

木马查杀深度剖析

目 录

- 一. 木马查杀深度剖析之扫盲篇..... (1)
- 二. 木马查杀深度剖析之进程篇（一）..... (3)
- 三. 木马查杀深度剖析之进程篇（二）..... (7)
- 四. 木马查杀深度剖析之进程篇（三）..... (12)
- 五. 木马查杀深度剖析之进程篇（四）..... (18)
- 六. 木马查杀深度剖析之自启动项篇（一）..... (24)
- 七. 木马查杀深度剖析之自启动项篇（二）..... (30)
- 八. 木马查杀深度剖析之自启动项篇（三）..... (35)
- 九. 木马查杀深度剖析之自启动项篇（四）..... (41)
- 十. 木马查杀深度剖析之文件篇（一）..... (46)
- 十一. 木马查杀深度剖析之文件篇（二）..... (52)
- 十二. 木马查杀深度剖析之文件篇（三）..... (61)

一、前言

在进入正文之前呢，容我先说几句废话。

在写此文之前，我曾写过一篇“高级木马的自我保护技术与查杀之策”，承蒙各位朋友的厚爱，被各网站所转载，亦曾有许多朋友来信求助。

由于该文是心血来潮后的随笔所写，所以并没有经过深思熟虑，也并没有脉络可循，至使有的朋友能看懂，有的却看不懂。而看得懂的呢，亦有很大一部分并不能真正的应用文中所提的查杀技巧。

故一直有心再写一篇文章，帮助深受木马毒害的朋友们了解木马是什么，又如何查杀。

本文适于下列人员阅读：

能熟练使用计算机的人

想自己动手解决问题的人

钱没有多到电脑一出问题就扔到电脑修理公司的人

适于下列人员参考：

电脑维修人员

微机管理人员

安全相关的程序开发人员

本文所要达到的目的是：

让菜鸟也可以了解系统安全自己来动手查毒杀马，由求助者变为帮助它人者。

在杀毒软件无能为力的情况下，借助本文的知识让你仍然能借助工具完成清理查杀的任务。

让您真正的明白一些看似神秘的、高深莫测的专用术语与技术的内幕。

让您了解 Windows 系统的底层知识。

本文以手动查杀为主，辅以必要的工具，文中用到的工具与测试用的仿木马小程序都提供了下载地址，供朋友们下载试用。

本文作者为 MuseHero，您可在网络上任意转载，但请注明出处、作者并保持文章的完整性，谢谢。

二、木马知识扫盲篇

什么是木马呢？木马只是一类程序的名字，为实现特殊目的而制作并偷偷植入目的计算机中的程序的统称。

其名字源于古希腊神话《木马屠城记》，全名为“特洛伊木马”有兴趣的可以翻看相关资料，很不错的一篇故事。

那什么又是程序呢？呵，即使是扫盲篇，我们就不能回避这些让人头都大的问题，先从最基本的开始，熟悉计算机的朋友们可以直接跳到第三章去。

我们先了解一些基本的概念：

概念一：计算机。就是我们的电脑了，指的就是一堆硬件，显示器、主机、键盘、鼠标等等，当然了，拆开主机还有一堆零碎。我们只需要了解其中几个后面会提到的就行了。

CPU：计算机的指令处理单元。所有的工作都是由它来做的，同一个 CPU 同一时间只能处理一条指令，就是说一旦 CPU 被某一程序占用，那在同一时刻内其它程序就肯定是空闲等待状态（双核的可以同时执行两条，以此类推）。

内存：内部存储器。CPU 执行的指令都是由内存中读取的，所以，一个程序要运行首先要装入到内存中。

磁盘：外部存储器。存放文件数据的地方。计算机中所有的数据平时都是存在这里的，只有在需要执行或查看时，才读取到内存中。

但只有硬件，是不能使用的，还需要软件的配合，我们买电脑时电脑中大都安装了某一操作系统，主流的就是 MSWindows 系统了（还有其它的操作系统如 Linux 等，这里不做讨论）。这就是我们需要了解的也是以后要频繁出现的第二个概念了“系统”。

概念二：系统。系统其实在大多数情况下都指的是操作系统，也就是帮我们使用与管理硬件的软件系统，当我们按下机箱电源后，即由主板上的 BIOS 检测硬件、再交由硬盘中的引导程序启动操作系统，然后就出

现了我们所熟悉的 Windows 主窗口（计算机在启动时做了些什么，我们会在后面详细的讲解），而以后我们的所有工作，看电影、听歌曲、玩游戏、上网浏览等等都是在这个窗口中进行。操作系统维持这个窗口的显示及一些常用功能的完成，而这些功能是由一系列的程序来实现的，这又引出了我们的第三个概念“程序”

概念三：程序。程序是什么呢？程序就是一组指令执行序列。呵，有点专业了是不？什么又叫指令执行序列呢？

举例来说：我管理着一个工厂，正常情况下我让工人们按这样来工作“先去原料仓库取原料——→ 进行初步加工——→ 进行精细加工——→ 进行零配件组装——→ 验收——→ 合格则送入成品库——→ 不合格则销毁”。这就是一个指令执行序列，当情况为正常时，工人们执行的是这个序列也就是这个“程序”。而在旺销的季节呢，我还有第二个程序“取原料——→ 进行初步加工——→ 组装——→ 进成品库”。当情况为供不应求时，我就执行第二个程序。还有第三个、第四个……等等，在不同的情况下，我让工人们执行不同的程序。

计算机程序也是一样的，跟据用户要求的不同，执行不同的指令序列，比如您要画图，你可能会这么操作“点击开始菜单——→ 选择所有程序——→ 在所有的程序中选择附件——→ 再在附件中选择画图”，这就是你向操作系统下达的指令。操作系统得到您的指令后，则会执行画图程序，也就是实现画图目的的一系列指令，而这些画图的指令储存在“mspaint.exe”文件中。操作系统会将 mspaint.exe 中的指令装入到内存中交由 CPU 开始一条条的执行。

总结一下：程序是什么呢？说白了就是一个计划书，里面记载了先做什么后做什么。好的程序是好的计划，坏的程序就是坏的计划，比如：“收集炸药——→ 买雷管——→ 制成炸弹——→ 放到张三的床下面——→ 引爆”，这就是一个坏计划，相对于计算机来说就是一个坏程序。坏程序有破坏作用吗？也就是说你写了个爆了张三的计划书就可以炸了张三吗？当然不可以，只有计划是不行的，还要去执行才能真正起到作用。所以，一个木马程序仅仅是存在于您的计算机中时并不可怕，可怕的是它执行起来。程序执行起来是什么呢？那就是我们要说的第四个概念“进程”——→ 执行中的程序。

概念四：进程。程序一旦进入内存中开始执行，就叫做进程（进程其实就是容纳指令与资源的容器）。也就是说，他已经开始工作了，开始买炸药、制炸弹了，等执行到引爆那一条指令时，张三也就完蛋了。所以，检查可疑进程，就是查杀木马的关键环节，也是重要的手段与方法。找到木马的进程，并结束它，在它执行到引爆这条指令之前，就停止它的执行，并将它的程序自计算机中删除掉，就是我们所要达到的目的。

说到这里，细心的读者们可能想到了一个问题，画图程序的执行，是因为我们向操作系统下达了画图的指令，所以操作系统才调入画图程序开始执行画图的指令序列。但是木马程序又是谁给操作系统下达的指令让木马的工作计划得到执行的呢？这就是我们下面要说的第五个概念“自启动程序”

概念五：自启动程序。顾名思义，自启动程序，也就是不用您自己动手它自己就可以启动起来开始执行的程序。这是些什么程序呢？为什么要允许程序这样做呢？难道是专为木马准备的？呵，当然不是。自启动程序有两大类，一是系统需要的；二是用户需要的。

系统需要的，是因为有些工作是无须经过用户同意，必须去做的。比如，鼠标驱动。为了能让用户使用鼠标，系统要自动加载鼠标驱动程序并执行。

用户需要的，一些重复性工作可能用户想让系统自动去做，而不是每天每时的重复这份工作。比如：用户可以设定多长时间无操作，就自动运行屏幕保护程序以便保护屏幕不被烧坏。这显然必须要屏幕保护程序能自动的运行，用户是不可能每次手动去执行这个程序的。

而木马就是利用了这些本来是为系统或用户提供方便的手段，来实现自动运行它们自己的目的。如：木马用自己来取代屏幕保护程序，这样，长时间无操作时，运行的就不再是屏幕保护程序而是木马了。

在系统中有很多地方或方法是可以让程序自动运行起来的，这个我们在后面将一一讲述。而清掉自启动项，让木马程序得不到执行，显然也就成为了我们查杀木马的重要手段之一。

而系统又是如何知道，哪一个程序应该在开机后就自动运行，无须等用户来操作呢？朋友们应该能猜到，系统肯定是把这些需要自动运行的程序都记录到了某一个地方，记录到哪里了呢？这就是我们要讲的第六个概念“注册表”

概念六：注册表。注册表是什么呢？是系统记录信息用的一个数据库，举例来说它就像公司档案柜一样，发工资时，财务人员要查阅档案，以确定哪个员工应该发多少钱。就像系统启动时查阅注册表，确定哪个程序应该自动启动起来一样。

这是一个非常庞大的数据中心，系统的关键信息都记录到那里，与现实中我们的档案柜一样，都很重要，一旦损坏将造成无可挽回的后果，所以微软公司不建议直接对注册表进行操作。当然了，木马的作者通常是不会理会微软的建议的。所以，木马通常都会将自己写入到注册表的某个自启动项中，以便开机时自动运行，而无须经过用户的同意。

讲到这里，又有一个不容回避的问题出现了，上面我们说了，木马程序其实就是一个计划书，而为了执行这份计划书，木马需要把自己写到注册表中的自启动程序序列中去。而问题就是，这份计划书是如何来的呢？如果得不到执行，它又如何将自己写到注册表的自启动程序中去呢？不放到自启动程序中它就得不到执行，而得不到执行，它也就无法起到作用也包括无法将自己写到注册表中去……晕了，这成了先有鸡还是先有蛋的问题了。那么木马是如何进入我们的计算机并获得首次执行的呢？

这就是我们要讲的第七个概念了“侵入”

概念七：侵入。侵入是什么呢？也就是侵略进入喽，侵略需要被侵略者同意吗？当然不需要。将木马程序放入您的计算机，并让它得到首次执行的过程就是侵入。

侵入的方法有很多种，我们将在后面主动防御一章中详细讲解。因为，相信现在朋友们最关心的并不是如何不让他进来，而是……我的电脑中现在是不是正在执行着木马程序呢？已经执行到了哪一步呢？是否马上就要引爆了呢？我关机后下次还能打开不？

好，接下来，就让我们进入下一章，一起来看看，电脑中正在执行的木马程序“木马进程”。

木马查杀深度剖析之进程篇（一）

三、木马的查杀之进程篇

1、进程的查看

进程，我们上面说过了，狭义上讲就是正在执行中的程序。那如何来查看系统中都有哪些程序正在执行呢？（先看下图 03-1:）



系统自带了一个“任务管理器”可以使我们看到系统中当前的进程，在桌面下方的任务栏按右键，选择“任务管理器”或同时按下“Ctrl Alt Del”三个键、或同时按下“CtrlShift ESC”三个键，都可以打开任务管理器程序，看到上面的窗口。

看到上面的图时，会不会有点发昏？20 个进程，哪个是好的哪个是坏的呢？上面的信息又都是些什么意思呢？

不要着急，让我来教教你怎么来看这些信息。

首先，显示哪些信息，是可以自由选择的，看到最上面的菜单没？就是“文件、选项、查看、关机、帮助”。依次选择“查看”——→“选择列”并勾选里面的相应项就可以显示相应的信息。

举报帖子

我们关心的是前五个，即：映像名称、PID、CPU、CPU 时间、内存使用，下面依次进行介绍。

1 映像名称：即进程所对应的同名程序名字。其中有两个是例外，“System”代表的是系统，并没有对应的同名程序；“System Idle Process”代表的是空闲进程，同样没有对应的同名程序，它占据了 CPU 的空闲时间。

我们可以依据此栏，来找到进程对应的程序文件。

1PID：英文缩写，即进程的 ID，是一个唯一的数字，唯一的代表一个进程。

我们可以依据此栏来区分进程，尤其是同名的进程，比如：SVCHOST.EXE 进程。

1CPU：即该进程当前消耗 CPU 的百分比，如果某个进程正在工作，那么 CPU 一列的数值就会是非零，工作量越大，其数值越高。例外的是“System Idle Process”进程，它的数值越高，说明当前的系统越是空闲，所有进程的 CPU 一栏的数值相加等于 100%CPU 占用。

我们可以依据此栏来查看，哪些进程正在工作，哪些进程是空闲的。上面我们说过，同一 CPU 在同一时间只能处理一个工作，所以如果某一进程长时间大量占用 CPU，那么将会导致其它进程得不到或得到很少的 CPU 时间来处理，使系统反应速度严重迟缓。这种情况的出现，通常是程序出现了问题，我们就要考虑结束掉霸占 CPU 不放的进程，并尽量查明原因。

1CPU 时间：自运行以来，该进程占用的全部 CPU 时间，此数值越大，代表该进程工作时间越长，注意，

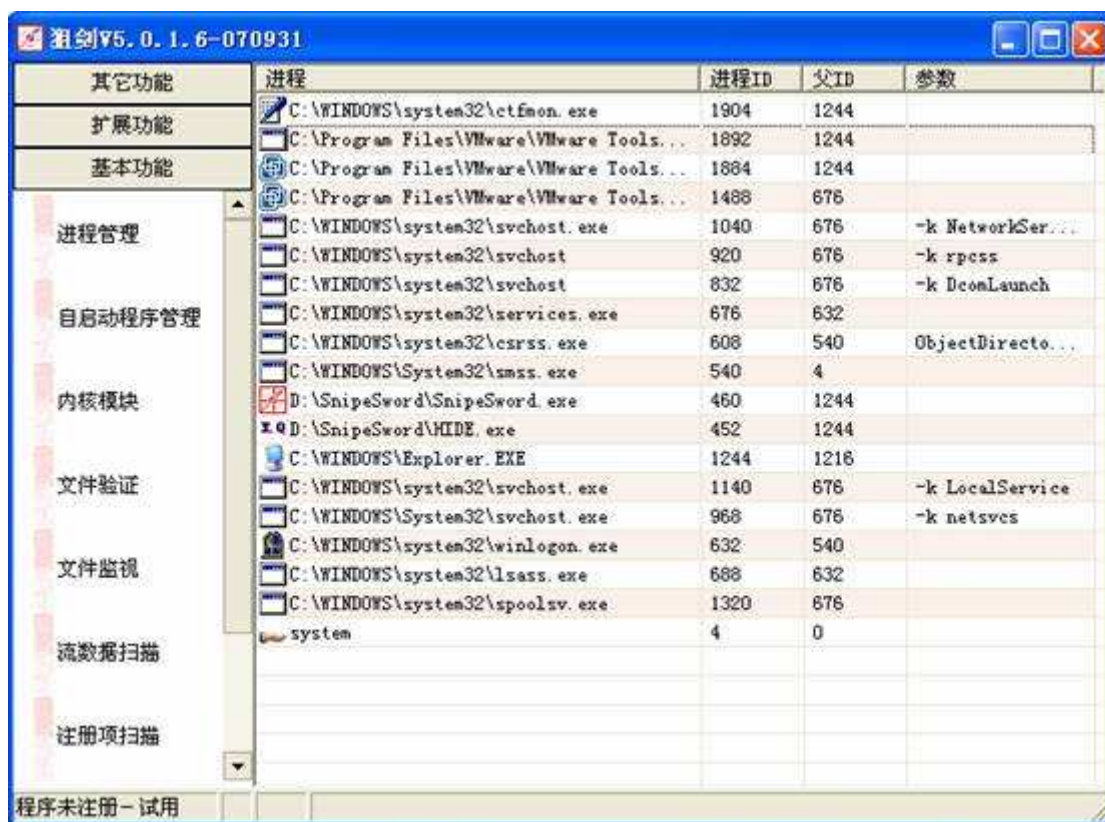
不是运行时间而是工作时间，如果运行后，进程只是等待，并没有工作，那么此时间也会很小。例外的仍然是“System Idle Process”进程，它占据了 CPU 全部的空闲时间。

我们可以据此来判断，哪些进程在一直的工作，而这一直工作的进程是不是应该工作的正常进程。比如我们一直在画图，那画图程序 mspaint.exe 的进程 CPU 时间很长，那就是很正常的；而如果我们在听音乐，从来没画过图，但这里却显示画图程序用了大量的 CPU 时间，那很可能就是某个黑客或木马或其它进程正在偷偷的使用画图程序呢。

1 内存使用：标明了该进程使用的内存数量，要知道，系统中的内存数量是有限的，当某一个进程大量占用内存时，就会导致系统或其它进程可用内存量减少，而至使系统运行速度减慢。

在某些情况下，可以依此来找出系统变慢的原因，并尽量使用占用内存小的程序来提高系统的性能。虽然现在机器的配置都高了，内存也都大了，但在同等条件下，优先考虑使用占用资源少的程序也是有必要的。

要在这些进程中区分哪些是系统进程，哪些不是，用任务管理器显然很难做到，最好的办法是借助专业工具。（看下图：03-2）



上图是用反黑工具狙剑查看进程时的样子，狙剑程序的下载地址：

<http://www.zhulinfeng.com/Download/SnipeSword.rar> 下载后解压缩，运行里面的 SnipeSword.exe 启动狙剑，并选择“进程管理”

就可以打开进程管理页面，与任务管理器相比，首先映像文件带了路径，使您可以直接定位到程序文件。

其次，多了父 ID 一栏，这个标明了该进程是由谁启动的，比如：smss.exe 的进程 ID 是 540，父 ID 是 4 那么，我们就可以知道，smss.exe 是由进程 ID 为 4 的 system 进程启动的，而 smss.exe 又启动了 csrss.exe 与 winlogon.exe，接着由 winlogon.exe 启动了 lsass.exe 与 services.exe。注意，有的进程并没有看到父进程，比如：Explorer.exe 它的父 ID 是：1216 但进程中并没有进程 ID 为 1216 的进程，这是因为系统在初始化时还有一个程序会自动运行，那就是 Userinit.exe 这个程序负责用户初始化工作，并负责启动 Explorer.exe，而工作完成后，它就退出运行了，所以进程中并没有它。最近有个木马是通过感染 Userinit.exe 来启动的，由于 Userinit.exe 启动后就退出了，所以通过查看进程很难发现问题，这问题我们将在“自启动项”一篇中详细讲解。

以上就是系统启动的顺序，从 System 开始，依次启动了几个进程，正常情况下，上面这几个进程都是

系统关键进程，结束它们的运行将导致系统出现问题。

有很多朋友对开机以后系统都做了些什么比较感兴趣，而这对系统不能启动时的故障查明也有帮助，可以通过判断系统停在了哪个环节来断定是哪个部分出了问题，在这里，我就对系统的启动过程进行一下简单的说明：

系统的启动过程：

1、当你按下开机键时，最早是由主板 BIOS 芯片中的 BIOS 程序来执行硬件检测任务的，如果检查过程中发现关键硬件的故障，就会发出特定的响声通知用户，并停止启动。而 BIOS 程序是哪里来的呢？这是在主板出厂时固化在芯片中的一段程序。

2、当硬件没有问题时，BIOS 程序就会读入硬盘的主引导记录，将下面的任务交给主引导记录代码去完成。而主引导记录又是哪里来的呢？这是在安装操作系统时对磁盘进行分区格式化操作时，写到硬盘中的。如果找不到主引导记录，会出错停止，告诉你这是非法的系统启动盘。

3、主引导记录代码的工作是读入磁盘主分区的根目录，在里面读出 Ntldr 文件，并装入内存，然后将控制权交给它。注意看一下，你的硬盘根目录下是不是有个 Ntldr 文件？这个文件的属性是隐藏、系统，所以查看时要选择查看所有，且不隐藏被保护的系统文件才可看到。这个文件是哪里来的？呵，当然是安装操作系统时拷贝到硬盘上的，下面提到的文件都是在安装操作系统时拷上去的。如果没能找到 Ntldr 文件，则会停止启动，显示 Ntldr 没找到的错误信息。

4、Ntldr 又做了些什么呢？它会将系统由原来的 16 位实模式切换到 32 位保护模式或 64 位长模式。它的工作是读取根目录下的 Boot.ini 文件，显然引导菜单，在多操作系统的计算机中，可以看到这个菜单。接着它会清屏，并在 Win2000 下显示一个黑白的进度条，在 XP 下显示 XP 的标志图同时显示下面不断滚动的蓝色进度条，提示你它正在加载一些重要的文件。它在加载什么呢？它首先会加载 Ntoskrnl.exe、Hal.dll，如果这两个文件找不到会出错停机，并显示找不到相应文件的信息。接着它读入注册表的 SYSTEM 键文件，从中找出自动启动的各类驱动程序，这是很关键的，因为有些内核级的木马就是在这时启动的。每加载一个屏幕上的进度条就滚动一下子。中间如果某个驱动出问题，也可能会导致系统蓝屏崩溃。

5、接下来的工作由 Ntoskrnl.exe（或 Ntkrnlpa.exe）来进行，这是内核程序，它做的工作实在是太多了，这里就不再细说了。它的最后一步工作就是创建会话管理子系统，也就是我们上面说过的，由 System 进程创建的 Smss.exe 进程。

6、Smss.exe 进程负责创建用户模式环境，由用户模式环境向 Windows 提供可视的窗口界面。

它会运行 BootExecute 中定义的程序，正常情况下是 Autochk，一个检查磁盘的程序。但有些杀毒软件会把自己的程序加到这里，来实现引导时杀毒，如果您的系统安装了江民类的杀毒软件，那么此时就会执行它的引导期杀毒程序，就是进入系统前出现的蓝底蓝字的病毒扫描窗口。

Smss.exe 还会执行 SessionManager 中的文件删除、移动操作，也就是调用 API: MoveFileEx 并选择重启后移除文件的，就是在这个环节执行了。当前有很多号称可以删除一切文件的安全工具都使用了 MoveFileEx 来实现文件的删除，但是现在我们可以知道了，它的文件删除是在这个阶段执行的，而这时驱动程序已经加载了，所以用它们来清除驱动级的木马显然是不胜任的。

创建附加的页面文件。

加载 Win32k.sys，这个东西又是做什么的呢？这是一个内核模式的系统驱动程序，它负责了窗口的显示、屏幕的输入、鼠标键盘和其它设备的输入及消息的传递等。所以也是由 Win32k.sys 将显示器的分辨率设置为默认值的，也就是这个时候，咱们的计算机屏幕才真正的细致起来，在此以前都是 VGA 模式，当然了视频驱动是上面装载驱动程序时就已经加载了的，现在只是起到作用而已。

再然后呢，就是启动我们上面说过的那两个进程了。就是 Csrss.exe 与 Winlogon.exe 进程。

启动完这两个进程后，Smss.exe 就进入了无限的等待，它在等什么呢？它在等它创建的 Csrss.exe 与 Winlogon.exe，等着看这两个进程什么时候死掉，一旦他们中有死掉的，Smss.exe 马上罢工，让系统彻底崩溃。（在 XP 以后 Csrss 的死亡是由内核使系统崩溃的，而不是 Smss.exe），所以千万不要结束系统进程。

Csrss.exe 是做什么的呢？它负责的工作是创建或删除进程、线程，控制台与虚拟 DOS 机的支持等。它到此就开始工作了，不再参与后面的启动过程。但是 Winlogon.exe 还有很多工作要做呢，我们接下来看看后面的启动过程。

7、Winlogon.exe 是做什么的呢？看它的名字应该看出个大概了吧。是的，它是与登录相关的，但现在还不到显示登录窗口的时候，它还要先启动 Services.exe 及 Lsass.exe 进程，然后读取注册表 GinaDLL 中标明的 DLL，由这个 DLL 来显示一个登录对话框，也就是我们在进入系统时输入用户名与口令的窗口。

为什么要先启动 Lsass.exe 呢？因为，这是本地安全认证子系统，负责的就是本机系统的安全，用户名与口令的验证工作是由它来进行的。

还有一个我们上面提到过的进程也是这个时候由 Winlogon.exe 来启动的，是哪一个呢？就是那个 Userinit.exe，这是在用户登录进系统后，Winlogon.exe 启动此进程来进行用户初始化。你也可以自己加一个程序与 Userinit.exe 放在一起，那么，在这个时候 Winlogon.exe 会将那一位置的所有程序都启动起来。

当然了，相信你也想到了，这个还有那个 GinaDLL 也就成了木马启动的一个可选位置。

8、最后，由 Winlogon.exe 启动的 Services.exe 开始加载标明为自启动的各个服务，及标明为手动的却是有必要加载的服务（它所做的工作我们在后面细讲）。

9、而 Userinit.exe 呢，它在完成用户初始化后，就启动了 Explorer.exe，并功成身退。

10、最后，Explorer.exe 就成了我们的服务员，等待在那里静候我们的指令，听从我们的吩咐，进行相关程序的启动与功能的处理。

木马查杀深度剖析之进程篇(二)

接着上面的一篇来讲：

看完系统的启动过程后，我们再回过头去看那个进程图，是不是明白了很多呢？除掉系统启动环节中启动的进程外，我们再看其它的进程，注意看一下儿，剩余的进程是不是都是由 Services.exe 或 Explorer.exe 中的一个启动的。

Explorer.exe 是系统的 Shell 程序，响应用户的请求，并执行对应的程序的工作就是由它来完成的，比如上面我们说过的，您想画图时，将画图程序启动起来的就是这个程序。你想上网时，将浏览器启动起来的也是它。当然了，一些用户的自动任务也是由它来完成启动过程的。（这些我们将在下面的自启动程序一章中详细讲解）

Services.exe 是系统的服务控制管理器，由它启动的进程称为“服务”，是一组特殊的进程。此类进程是开机自动运行的，不依赖于用户的交互，说白了就是不用您管，它自己就会运行并且开始自己的工作，工作过程也无须您的干预。

说到这里，您可能会想，这不就是木马所需要的么？不错，有很多木马是以服务进程的形式存在的。那我们可以管理这些服务吗？答案是肯定的，我们的计算机，我们当然有权力管理了。看下面的图 03-3：



在桌面上“我的电脑”上面按右键，选择“管理”就可以打开如上图所示的窗口。依次选择：服务与应用程序——> 服务 就可以看到上图右侧的服务列表。

最左侧的一列是服务的名称，需要注意的是这里列出的并非是对应的程序文件的名称，而仅仅是服务本身的名称。那我们又怎么知道这个服务对应的是哪个程序文件呢？如果我们想停止这个服务，并删除对应的程序文件，我们应该怎么做呢？

再看下图 03-4：



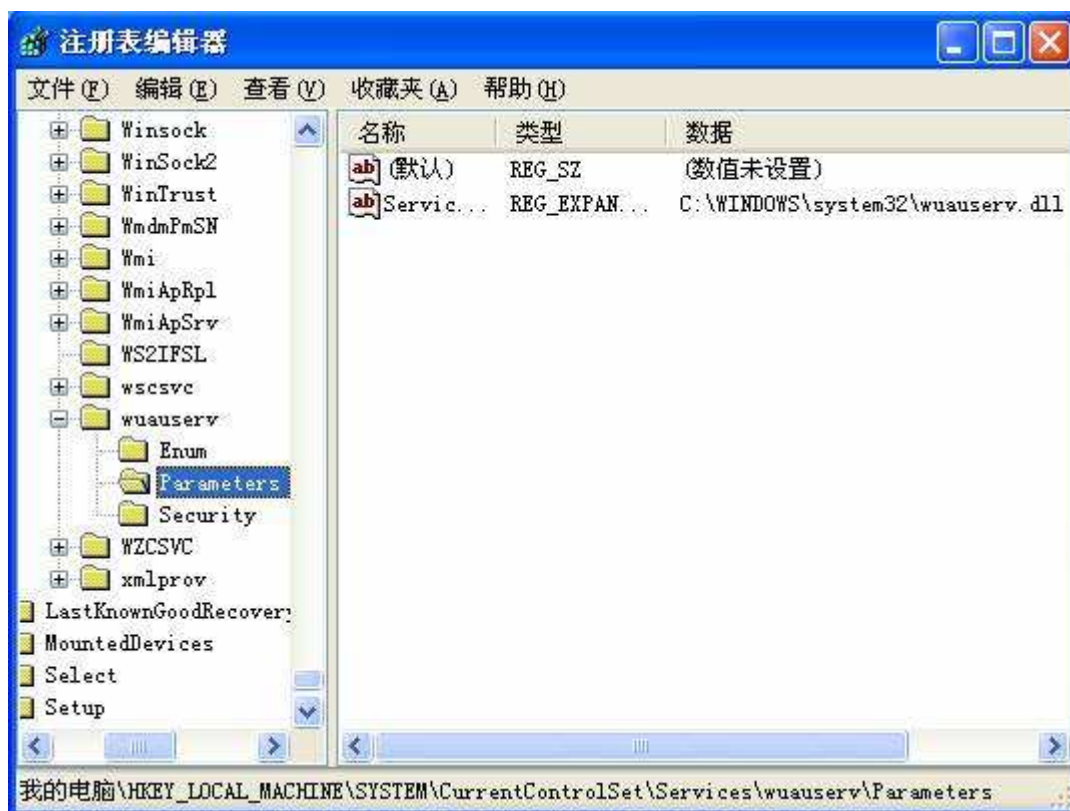
在某一个服务上面按鼠标右键，选择“属性”，就可以打开此服务对应的属性页面，如上图所示。在这里，我们可以看到一些有用的信息，并对此服务进行启动、停止、禁用等操作。

最上面的是服务名称，第二列是显示名称，第三列是关于服务的描述，也就是这个服务是做什么用的。当然了，我们不能迷信这里的描述，因为木马可以自行设定这个描述。再下面就是可执行文件的路径，也就是对应的进程与程序文件。这时我们再返回头去看 03-2 那个图，看一看进程 PID 为 968 的进程是不是就是这个“svchost.exe - knetsvcs” 进程呢？

这时再注意一下，又会发现问题，首先，在 03-2 图中，Services.exe 启动的服务一共是 7 个，而在图 03-03 中，已经启动的服务却有很多，远不止 7 个，再挨个儿的查看每个服务的属性，又会发现，有很多服务名称虽然不同，但对应的可执行文件路径（进程）都是一样的，都是“svchost.exe - knetsvcs”进程，这又是咋回事呢？这里我们要解释一下 Svchost.exe 这个进程了，这是一个特殊的进程。

从图 03-1 中我们可以看到名字为“SVCHOST.EXE”进程一共是五个，我们没办法区分它们有什么不同。而在图 03-2 中，我们再查看这五个 Svchost.exe 进程，却可以从参数上来区分它们了，虽然进程名字相同，但它们的参数却各不相同。对了，这就是区分他们的方法了，Svchost.exe 是一个服务的宿主程序或者叫容器程序，它的里面是一组服务，而参数就标明了这一组服务的类型。如果英文不是太差，那么从它的参数上就可以看明白这一组服务是做什么用的了。

当然了，我们是来查杀木马的，当然不会满足于仅仅知道这是一组服务，这一组中是不是会藏着一个木马呢？所以，我们还要知道这一组服务都分别是哪一个，怎么来查看呢？再来看下图：03-5



上面的是什么呢？上面的就是我们在第一章中所提到的“注册表”。依次选择——> 开始——> 运行——> 输入“Regedit.exe”——> 确定。

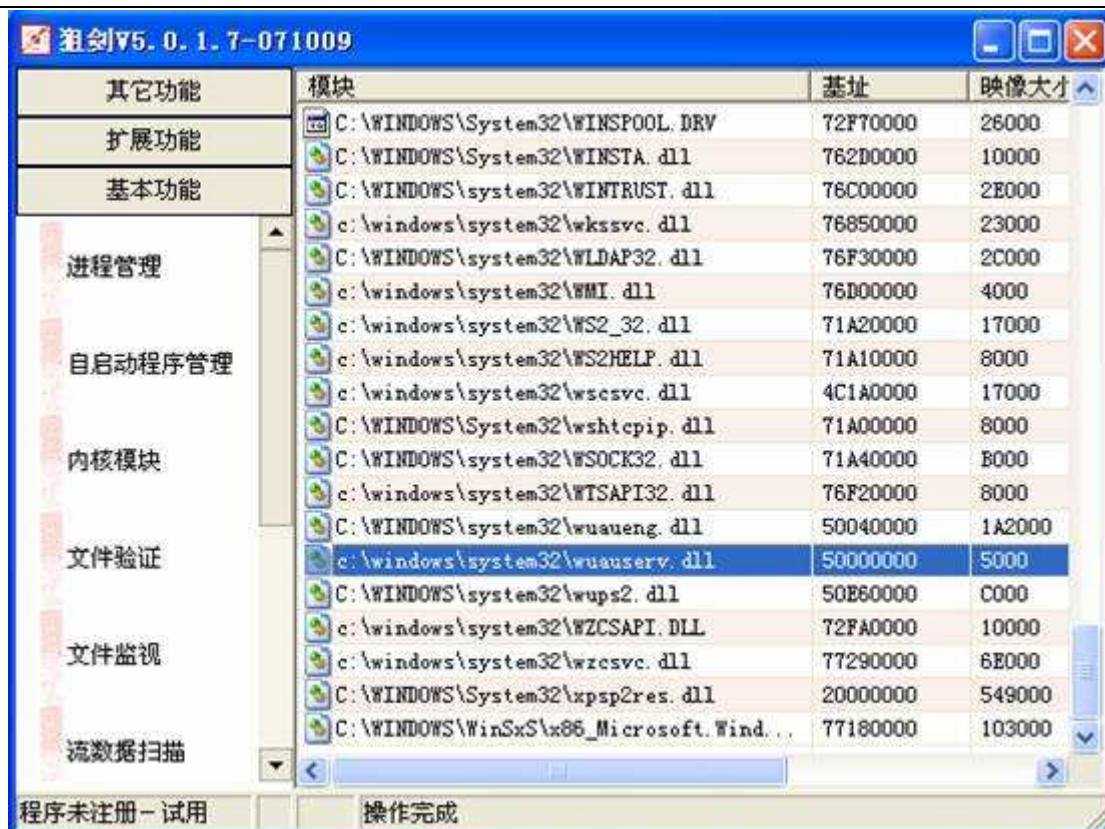
就可以调用系统自带的注册表编辑器来打开注册表，再依次展开：

HKEY_LOCAL_MACHINE——>SYSTEM——>CurrentControlSet——>Services，还记得上面 03-4 图中的服务名称么？对了就是那个“wuauiserv”，在 Services 键下打开如上图所示的“wuauiserv”再展开，选中其下面的“Parmeters”，看右边的窗口中，是不是找到了此服务所对应的程序文件了？就是那个 C:\WINDOWS\system32\wuauiserv.dll 喽。

每一个 Svchost.exe 中的服务都可以通过这种方式找到其对应的程序文件。

看到这里，朋友们是不是头都大了？这么麻烦啊？查个进程居然这么麻烦，而且就算找到了，又怎么能知道 C:\WINDOWS\system32\wuauiserv.dll 是正常的程序还是木马呢？

呵，不要着急，也不要怕。上面不是在教给你知识么，而且在当年，没有专业工具之前，我们可都是这么一个个来找的。当然了，现在有了各种专业工具，的确是没必要非手动去找了。



在狙剑的进程管理列表中（图 03-2），选中 Svchost.exe，然后按鼠标右键——→ 选择“查看模块”，就可以得到上图中的列表，注意被蓝条选中的那个是不是就是费半天劲所找到的那个：

C:\WINDOWS\system32\wuauerv.dll 呢？

找是找到了，可如何判断这是不是系统的服务模块呢？这一点微软为我们想的很周到，他将大多数的系统文件都进行了数字签名，我们只需要检查一下这个文件有没有系统文件的数字签名，就可以准确的判断，这是不是一个系统文件。

如何检查呢？在列表中按右键，选择“隐藏微软签名项”就可以将所有系统文件隐藏起来，剩下的就是非系统的文件啦，简单不？

数字签名验证还可以应用在对进程的判断中，在图 03-2 中，列表中共有 19 个进程，这些个进程中哪些是系统的哪些又不是呢？我们仍然可以利用数字签名来判断，在列表中按右键，选择“隐藏微软签名项”就可以将所有系统进程藏起来，剩下的就是非系统的进程。

上面也说过，一共有 5 个 SVCHOST.exe 进程，那我们是不是需要在每个 SVCHOST.exe 进程上都重复一遍，查看模块——→ 隐藏微软签名项的过程呢？答案是不需要，在图 03-2 的进程列表中，按右键，里面的一个“扫描无微软签名模块”的选项，点击它一下，就可以列出所有进程中所有没有签名的模块了。

还有一个问题是，如果手头没有工具怎么办呢？就没有办法查看这些进程中的模块了吗？有的，仍然有办法，看下图 03-7：



依次选择“开始”——>“所有程序”——>“附件”——>“系统工具”——>“系统信息”就可以打开上面的窗口。在窗口中选择“加载的模块”就可以看到左边窗口中的信息，看到没？我们找的那个wuauerv就在里面。这里面列出了所有进程加载的所有模块，可以参考，但也仅仅是参考，更多的情况仍然尽量地使用专业工具来检查，会更方便更准确一些。

另外，有很多木马是可以隐藏的，对于这一类木马，用系统自带的工具是检查不出的。注意看上面的几张图，发现没有，图 03-2 比图 03-1 中的进程多出来了一个 Hide.exe 这就是下面我们要讲的“进程的隐藏了”。也许有朋友会说，是不是取图时这个进程已经关掉了呢？呵，没关系，我这里提供了 Hide.exe 程序的样本，您可以下载后运行一下试试，看是否可以在任务管理器的进程列表中看到它。

下载地址：<http://www.zhulinfeng.com/bbs/dispbbs.asp?BoardID=3&ID=78&replyID=&skin=1>

里面还附了隐藏进程的源代码，懂编程的朋友们可以参考一下，看一看隐藏进程是如何的简单。

木马查杀深度剖析之进程篇(三)

2、进程的隐藏与自我保护

我们通过上面的讲解已经知道，进程是一个程序运行所必须的，因此检查进程也就成了查杀木马的关键环节，我们知道这一点，木马的作者们当然更知道，所以，如何隐藏自己的进程，就成了养马人处心积虑要实现的。

说到这里，有懂的朋友们可能要笑了，心里更可能在说“连无进程木马都不知道，还好意思在这里显摆呢”。

一个程序可以无进程么？可以吗？真的可以吗？我可以很负责的告诉您，在 Windows 系统下一个程序一定、确定以及肯定的会有一个进程，没有进程是不可能的。

那所谓的“无进程木马”又是怎么回事儿呢？

第一种无进程木马是 DLL 注入型木马：

什么是 DLL 呢？DLL 是动态链接库，当某一进程需要实现某一功能时，此功能可能是放在某一动态链接库文件中的，所以，当进程需要使用时就要将动态库文件加载到自己的进程中。举例来说，如果进程就是

一个工厂，那么，调用 DLL 就是将工厂的某一部分工作外包给了 DLL 去做，而工作地点就在自己的工厂内。明白了吗？进程就是工作的厂地，即使外包出去了，也是需要地方干活的。“加载到自己的进程中”这一句是关键啊，并不是没有进程，而是用了其它程序的进程。像我们上面找了半天的那个服务：

C:\WINDOWS\system32\wuauiserv.dll，注意一下文件的扩展名，不就是 DLL 么，这就是一个典型的 DLL 文件，如果这是一个木马文件，那么，这就是一个典型的无进程木马，因为他没有自己的进程，我们在进程列表中看到是 Svchost.exe 的进程。这个 DLL 是利用服务来加载的，在注册表中还有很多位置可以让一个 DLL 加载到其它进程中，这在后面自启动程序一节中我们要讲解。

但是，通过注册表来加载是可以的，但却不是唯一的，还可以通过另一个进程，来打开现有的进程，来将 DLL 注入到被打开的正常进程中，然后，执行注入的进程退出，这样，在进程列表中仍然看不到木马的进程。

第二种无进程木马是线程注入型的木马。

什么是线程呢？上面说了程序就像一个工厂工人工作的流程表，而进程就是将流程真正的运行起来，正式的开始动手取原料→加工→组装→验收等工作。但是，我们需要明白的一点是，工人们是可以分成小组后同时进行几个环节的，第一组工人负责取原料、第二组工人负责加工、而第三组工人负责组装、第四组……这样，可以分工合作同时进行形成流水线。而这工厂里的一组一组工人，相对于进程与线程而言，就是一个进程中的一个线程。线程注入又是咋回事呢？线程注入，就是木马程序将一个恶意线程放到了正常进程的线程序列中去执行，就像在工厂中多增加了一组自己的工人负责制造炸弹，这一组工人与其它正常的几组工人没有什么关系，但却借用了人家的工厂去从事着非法的勾当。

线程注入与 DLL 注入的区别是，线程注入只是增加了一组工人，这组工人是在以工厂的名义在工作，对外的名义也是工厂的名字，出了问题是由工厂负责的。而 DLL 注入呢，是外包，可能会增加一组工人也可能会增加多组，是以 DLL 自己的名义在工厂内工作的，出了问题是由 DLL 来负责的。当然了，如果问题大了，工厂也会受牵连的。看下面的图 03-8：



在狙剑的进程列表中，选中一个进程按鼠标右键，选择“查看线程”就可以查看此进程中的每一个线程，也就是查看每组工人的情况，还可以对某个线程执行暂停运行、结束运行等操作。

我们注意看上面的线程列表，最后一列，显示了一些 DLL 的名字，还有一个是 EXPLORER.EXE，EXPLORER.EXE 就是进程名字，我查看的就是 EXPLORER.EXE 进程的线程列表。

明白我上面说的区别了么？ntdll 有三组线程在工作，那是 ntdll 在完成外包业务所需要的自己的三组工人；EXPLORER.EXE 有一组，那是工厂自己的工人，如果存在线程注入，那么就会多出一个 EXPLORER.EXE 线程。但是需要注意的是，有多个 EXPLORER.EXE 并不代表就一定有线程注入，因为一个工厂也可以同时存在多条流水线。但只有一个 EXPLORER.EXE 那么通常情况下就是没有线程注入，因为一个进程肯定会有至少一个线程的存在，也就是说，一个工厂必定至少要有一组工人，皮包公司在 Windows 世界是不允许的。

了解了什么是 DLL 注入型木马、什么是线程注入型木马后，查杀他们显然就很容易了。

DLL 注入型的，查起来很简单，用上面说的方法“查看某一进程的模块列表——→再隐藏无微软签名的项”，就可以查出单个进程中被注入的模块；如果在全系统范围内查找，则使用“搜索无微软签名的模块”就可以将注入到其它进程中的模块找出来，而大多数工具都提供了“卸载模块”的功能，卸载后就可以删除了，或直接用“卸载后删除”的功能，但有一点是需要注意的，如果该木马使用了防卸载的技术，那么卸载此模块时就会发生异常，对此也不必担心，我们还有其它方法清掉它，比如：清除其自启动项，不让他有注入的机会；或直接删除其磁盘文件等。

相比较起来，查杀线程注入型的木马，就比较困难了，我们上面说了，线程注入的没有自己的文件，只是一段注入的代码。也就是说他只是混入工厂内的一组工人，并没有自己的工厂也没有自己招牌，想把他与正常的工人区分开，是很困难的。这里提供了一个线程注入的测试程序：

<http://www.zhulinfeng.com/bbs/dispbbs.asp?BoardID=3&ID=78&replyID=&skin=1>

先试一试看看效果，运行后，会打开一个窗口，但你却在进程列表中找到有新增加的进程，那窗口是如何来的呢？这就是由注入到 Explorer.exe 中的线程创建的。这时你再查看 Explorer.exe 的线程列表，会发现多出来了一个线程，而线程的“模块”那一列（图 03-8）标的名字就是 Explorer.exe，这时线程列表中已经有两个名字为 Explorer.exe 的线程了，但哪一个好的，哪一个是后来注入的呢？通常情况下看“基址”那一列（图 03-8），基址数值较大的那一个一般是启动起来较晚的，就是后来加入的了，选中那一个线程选择“结束线程”，发生了什么？是不是那个窗口被关闭了？（注意：如果是用狙剑查看的，那么在线程注入后，查看新线程时，要重新刷新进程列表，不然会发现不了新注入的线程）

当然了，上面只是做个试验，让朋友们亲身体会一下线程注入是怎么一回事儿。真正查杀起线程注入型的木马来，用上面的方法显然是不行的，首先，我们不知道他注入到了哪一个进程中。其次，即使一个进程中有多个以进程名为名字的线程我们也无法确定那多出来的线程就是木马线程。所以，我们仍然要从自启动项入手来禁止线程注入。

第三种无进程木马是纯驱动型木马

什么是驱动型木马呢？就是全部功能放到了驱动程序中去完成（当然了，纯驱动型的并不多见，大多驱动型木马都是配了一个程序。）

什么又叫驱动程序呢？驱动程序顾名思义就是驱使设备动起来的程序。^_^呵，可能不准确，但却很容易理解。其作用是让特殊的硬件和 Windows 操作系统可以交换数据，比如我们按下了键盘的 A 键，那键盘驱动就要告诉 Windows 系统“这家伙按下了 A 键，你看咋办吧”，它只是告诉一声，后面的工作就由系统来处理了，系统会根据不同的情况进行不同的处理，如果你是在打字，那就把 A 这个字符显示在你的输入页面中，如果你用的是五笔，显然直接显示个 A 是不行的，Windows 系统就会把你输入 A 的这个信息转给了输入法程序……最终实现你按 A 的目的。

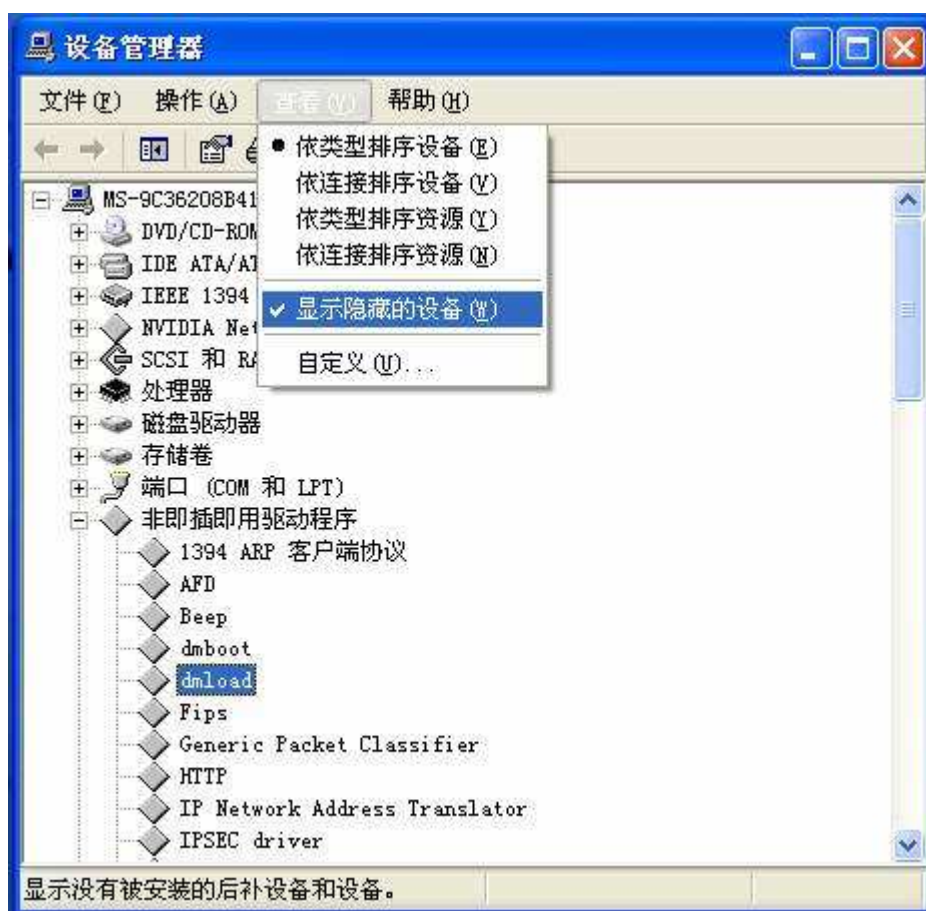
但驱动又怎么会与木马有了关系呢？这就要从头说起了，话说当年木马与驱动本来是分处于两个天地，互不相干的。但随着杀毒软件、安全工具等木马杀手对木马的围追堵截越来越是凶狠，木马终于感到有点穷途末路了，在最后关头，它盯上了驱动程序！为什么盯上了驱动呢？因为驱动程序由于其特殊性，使得它在系统中有着超越一切的权力，而且有着优先加载执行的优势，它是做为操作系统的一部分来运行的。详细情况涉及到系统权限的划分与优先级设定，有兴趣的可以查阅相关资料，这里只要明白一点就行“系统的权限分为两种 R0 与 R3，一般的程序全部为 R3 权限，其行为受到了很多限制；而操作系统与驱动则为 R0 权限，拥有对整个计算机的全部权力，可以为所欲为”。

举例来说，杀毒软件就像一个公司的人力部，对员工的行为进行考核，发现不好的就给予惩罚或辞退。如果木马总是隐藏在员工群里，那么就会受到严格的检查与监管，一旦被查出有问题就会被立即辞退。而操作系统就是公司的董事会或股东大会，驱动程序是董事或股东，有着超出人力部门监管的权力。

为了逃脱追杀，开始有木马以驱动程序的身份出现，当然了，木马这个驱动是没有硬件设备的，呵，如果一个木马它就附带一个设备，那么，哪怕它只是给一支鼠标，我也会冲天大喊“木马们，都到我这里来吧”。所以它们有一个好听且很容易听明白的名字“虚拟设备驱动程序”。

当木马成为公司的董事的时候，身为人力部门的安防程序开始感到郁闷，明明发现那家伙就有问题，但偏偏没权力辞退它。于是，杀毒软件开始谋求同样的身份，于是杀毒软件也开始挂驱动，使其有了董事的身份来兼人力监管，于是，新一轮的斗争开始了。而这一轮的斗争更是残酷，本来在员工层的斗争转到了高层，本来斗争的结果也就是员工被辞退，还不至于伤筋动骨涉及公司根本，但现在却是一不小心就让公司整个解体了。对应于我们这些可怜的用户来说，杀软与木马的斗争，一个不好惹来的就是蓝屏死机、系统崩溃。

OK，了解了驱动程序后，我们就要回到正题来了，我们如何知道系统加载了哪些驱动呢？又如何知道哪些驱动是正常的系统驱动，哪些是木马的驱动呢？看下图 03-9：



在桌面我的电脑图标上按右键，依次选择“属性——> 硬件——> 设备管理器”就可以打开上面的窗口了，再勾选上“查看——> 显示隐藏的设备”就可以看到“非即插即用驱动程序”。

除了非即插即用外，其它的同级项目都是您机器上的硬件及它们的信息、资源、驱动等，他们是根正苗红的正经设备。由于这些都与硬件有关，所以，木马动它们的可能性很小，毕竟木马也不想你的机器崩溃啊。因此，大多数木马驱动都隐藏在“非即插即用驱动程序”下面，而这里面的则是那些虚拟设备了（注：非即插即用驱动程序并非就是虚拟设备驱动程序，这涉及到以前老版系统中的一些个问题，这里不再细说，您只需要明白，在当前的情况下，这里面大多都是虚拟设备驱动程序就可以了。）

选中某一个项鼠标右键，再选属性，得到下图 03-10：



一看到这张图，你最感兴趣的一定是“停止”这个按钮，但我不得不很遗憾的告诉您“是否能停止驱动程序的运行，还要看这个驱动程序是否愿意停止运行”，是不是失望了？呵，应该是意料中的，如果木马可以这么轻易的停掉，它们还混什么呐？从这里不要奢望得到额外收获，我们只需要了解一点就够了，这个驱动的状态是“已经启动”。

接下来要看的是“驱动程序详细信息”这一栏。点一下，得到下面的这张图 03-11：



看到没？这里面的就是驱动程序对应的文件了，至于下面的提供商、版权什么的，参考一下也就得了，如果木马愿意，它可以说它是“天宫”出品，版权属于“玉皇大帝”，那这里会丝毫不折扣的给它显示出来，你说能信吗？

按上面说的方法，你可以查看所有系统中加载的驱动程序，但很显然，这么做的是很麻烦，那有没有省事点的方法呢？有，我们看下图 03-12：



看到上面这张图是不是有点眼熟？对了，就是上面我们查找模块时用到的图，打开方式同上，向上翻就可以看到了。蓝条选中的那个，不就是与图 03-11 中显示的是一个么。这里面列出了所有的驱动程序，不用一个一个找了。

到这里是不是有朋友有点火大了？明明有简单的法子，却带着我们绕了这么大一个圈子。呵，上图中的系统信息，其实也就是系统自带的一个工具，属于专业工具范围内的，且并不是太好使，木马想在这里隐藏信息很容易，而且禁止这个的运行也很简单。我们需要多了解几个法了，以备不时之需，另外，我们需要的是尽量多的学一些知识，这才是根本，了解了原理、懂得了知识，才能真正的成为高手，高手绝非只会使用工具。

说到工具，那我们再看看专业工具对驱动程序的显示：



打开狙剑——> 选择基本功能中的“内核模块”，看到没？dmload.sys 不就在这里么。这么多，又如何来区分哪个是正常的哪个是可疑的呢？很简单，方法与我们筛选进程是一样的，按右键——> 选择“隐藏微软签名项”，剩下的，就是非系统的驱动了。

这里同样提供了“停止运行”、“卸载模块”等功能，但我还得遗憾的告诉大家，卸载它也得它自己愿意才行，但想来木马是不会愿意被卸载吧，那我们就没办法了么？有，方法仍然与上面的相同，从自启动程序入手，禁止它的加载；或强制删除它的文件。

那驱动级木马是不是就是真正的无进程呢？进程列表中找不到任何一个驱动程序的进程，而它也不像模块与线程一样，在其它进程内能够找到，是不是真的可以无进程运行呢？当然不是，我前面已经说了，一个程序的运行，一定、确定以及肯定的会依托一个进程，那这些驱动在哪个进程里呢？那就是 System 进程，它们是做为系统的一部分来运行的。

木马查杀深度剖析之进程篇（四）

第四种无进程木马就是利用技术手段隐藏进程的木马

这显然就不属于无进程了，上面说的三种，的确是没有自己的进程，只是利用了其它的进程。而利用技术手段隐藏进程的木马，则是有自己的进程，但是如果你破解不了他的隐藏技术，那你就看不到它的进程。就像上面我们拿出来那个 HIDE.exe 一样，在任务管理器中看不到它的进程，而在狙剑中却能看到，这就是隐藏与破解隐藏后的结果了。

进程是如何隐藏的呢？这无可避免的会涉及到一些技术问题，下面我们尽量简单明了的说一说，详细的实现细节，请参阅相关资料。在这里，我所想达到的目的，并非是让您也同样写出一个能隐藏自身的木马，而是让您明白，什么样的手段可以隐藏木马的进程，木马又为什么会隐藏，而想破解这种解藏将它揪出来，又需要什么样的技术，好有针对性的选择一些专业工具。

同时，需要注意的是，隐藏进程的技术同样适用于隐藏 DLL 模块程序、隐藏驱动程序，下面为了描述方便，统一说为进程。

Windows 系统给我们的开发人员提供了几种列出系统中所有的进程、模块、与驱动程序的方法，最常见的也是最常用的方法就是调用系统 API：CreateToolHelp32Snapshot、EnumProcess、EnumProcessModules

等，如果您不是开发人员，您不用关心这几个是什么东西，只需要明白，他们是获取进程列表的第一层手段，我们调用这几个其实就是在告诉系统，我们需要进程列表，然后系统就会将列表返回给我们。

而这几个 API 在接到请求后又做了什么呢？他们会调用 ZwQuerySystemInformation，ZwQuerySystemInformation 会调用 KiSystemService 切入内核进入 R0 权限，然后自 SSDT 表中查取得 NtQuerySystemInformation 的地址，并调用其指向的实际代码，而 NtQuerySystemInformation 的作用则是自系统的数据结构中取相应的数据，再顺原路返回去。

在中间任何一个环节进行拦截都可以实现隐藏进程的目的，这种拦截有一个名字叫做“HOOK”，在切入内核进入 R0 权限前进行 HOOK，称为应用层 HOOK，而在之后进行 HOOK 则是内核 HOOK，后者需要用驱动才能实现了。

什么是 HOOK？什么是 SSDT？我们来举例说明：

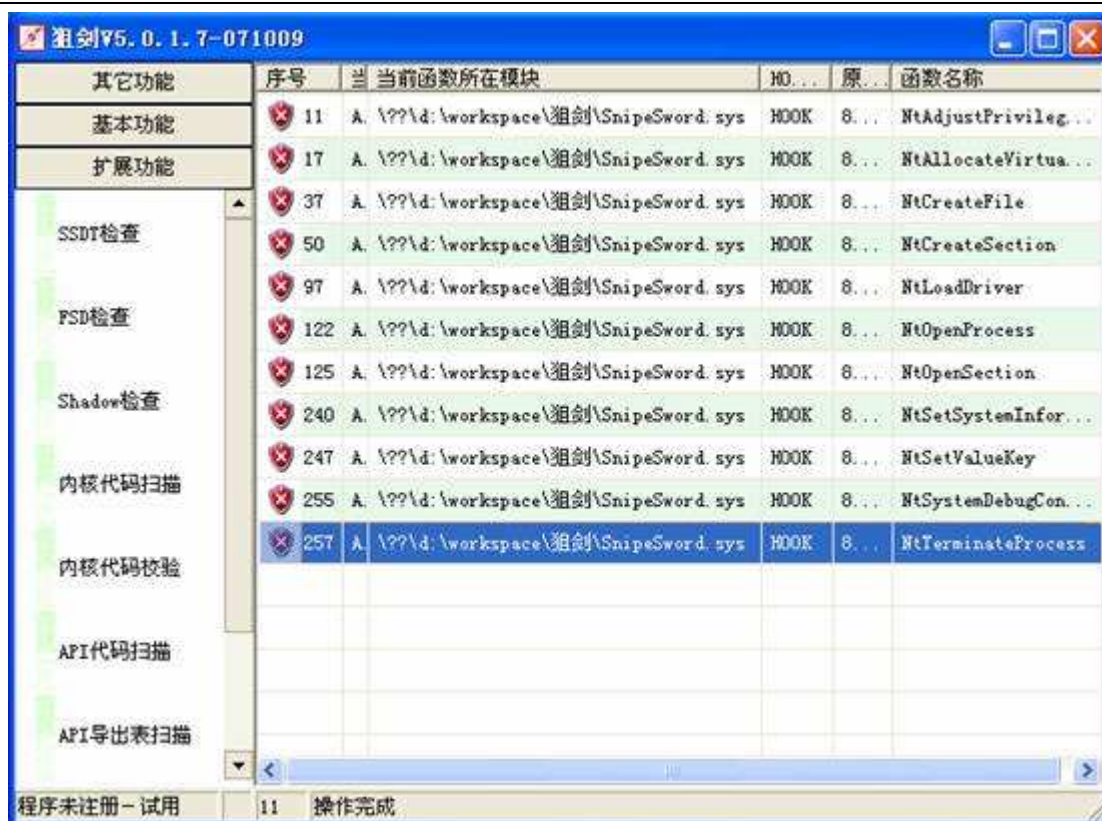
Windows 操作系统就像一个为我们管理电脑的服务公司，而他的工作机制是逐级上报，各负其责的，他派了一个服务人员时刻的跟着我们，看我们都有什么要求。

当我们想查看系统中都有什么进程时，我们会告诉服务员，我们想了解当前都有哪些进程，那么服务员就会把我们的要求报上去，报给谁呢？他要先知道哪个部门负责哪个工作才行，SSDT 表就是做这个的，SSDT 表就像是一个路标，指明了什么样的工作应该由哪一个部门负责处理。通过查表得知负责的部门后，工作就被移交到了那个部门，工作完成后，部门会把工作结果交回给服务员，服务员再交给我们，而我们也就得到我们想要的结果了。

应用层 HOOK 呢，就像是服务员被木马偷偷的替换了，当我们提出要求时呢，他会检查是否对他有害，或将对有害的信息去掉，比如我们想查看进程，那他在将结果交给我们时，却把木马的进程自结果中抹去了，这样，我们自然是看不到木马进程了。

而最常见的内核 HOOK，则是 HOOK-SSDT，上面说了 SSDT 就是一张表，标明了什么工作应该由什么部门负责。而 SSDT HOOK 也就是木马将表上的内容给改了，本来交给 A 部门负责的工作被改成了交给由木马负责，这样，服务员在上报我们的请求时，一查表发现查看进程的工作是由木马负责的，他就把这请求交给木马了，而木马呢？他可是知道表中原来的内容是什么的呀，他只是对请求进行一下过滤，发现没有对自己有害的，就直接转交给原部门了，对自己有害的，自然也就视情况滤掉或涂改了。

我们先看看 SSDT 到底是什么样子的，看下图（04-14）：



依次选择狙剑→扩展功能→SSDT 检查→筛选可疑项，就可以看到上图，那就是一个 SSDT 表，从左到右依次是序号、当前地址、所在模块、HOOK 类型、原地址、函数名。

上面都是狙剑自己对 SSDT 的 HOOK，当前安全软件很多也用到了 SSDT-HOOK 技术来实现对系统的安全防护。

例如上面的第 11 号函数，所在模块是“D:\workspace\狙剑\SnipeSword.sys”，HOOK 类型是“HOOK”，函数是“NtAdjustPrivilegesToken”。这是狙剑本身对 SSDT 的一个 HOOK，用的技术就是 HOOK，HOOK 的函数是用于增加进程权限的。HOOK 这个的目的是“在木马进程为自己增加权限的时候进行拦截，提醒用户注意”这在主动防御一章中会详细讲解。

每一个函数都实现了某一种功能，比如：结束进程是由 NtTerminateProcess 来完成的（上图中最后的那个 257 号函数），如果 HOOK 了这个，那么在进程结束前，就有机会更改结果了，可以拒绝被结束。看下图（03-15）



当你试图在任务管理器中结束狙剑的进程时，系统会拒绝你的操作，其实这并不是系统拒绝的，而是狙剑自己，由于狙剑 HOOK 了 SSDT，所以，结束进程的工作服务员都交给他去做了，他一旦发现结束的就是他自己，那他直接告诉服务员这个进程不能结束，然后服务员就把这个结果给我们了，其实我们的这个请求并没有真的到达应该送交的部门。

而上面提到的查看进程用的 NtQuerySystemInformation 也在这里面，注意找一找就会发现了，如果想隐藏进程就可以把这个给 HOOK 了。

需要特别注意一下子“HOOK 类型”，上面的显示是“HOOK”，还有一种是“Inline-HOOK”，什么是 Inline-HOOK 呢？

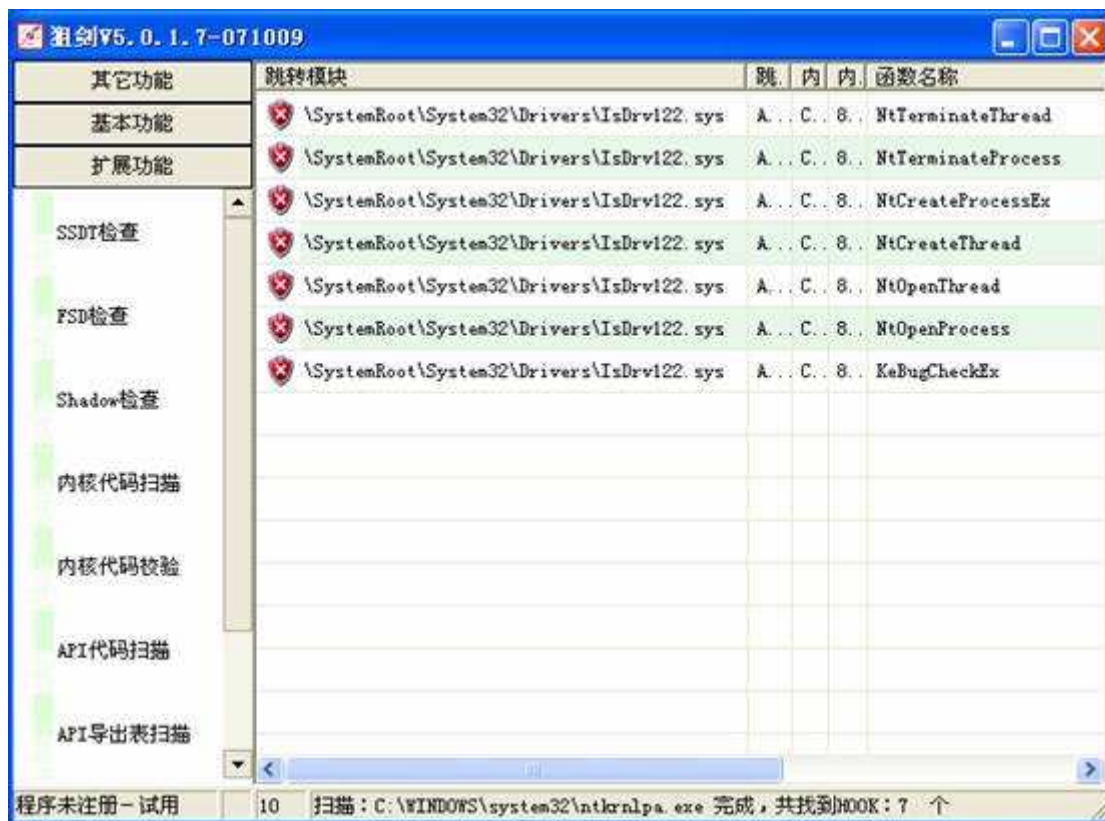
上面说过了，SSDT 就像是一个标明了什么工作由哪个部门来做的表，SSDT-HOOK 就是更改了这个表的指向。而 Inline-HOOK 呢？他并没有更改表的指向，查找进程的工作在表中仍然指向了负责查找进程的部门。但是呢，木马却把那个部门中的人员替换了，这样仍然能达到它的目的。

Inline-HOOK 更加的复杂、更加的邪恶、也更加的不稳定，而应用范围却是更加的广泛，查找起来更加的困难。要知道，对比 SSDT 表中的指向是否正确，是否指向了正确的部门，要简单一些，不就是一张表么，拿原来的表对比一下就知道了。但 Inine-HOOK 却是替换的公司人员，Windows 就像一个大公司，有成千上百的人，天知道他替换了哪一个，我们先看看下面的图（03-16）：



当试图结束 IceSword.exe 时，会出来上面的图，IceSword.exe 是另一款知名的安全程序“冰刃”的主程序，当您试图在任务管理器中结束 IceSword.exe 的进程时，就会出来“无法中止进程”的对话框。

但是，当你检查 SSDT 表时，却发现，它并没有对 SSDT 进行 HOOK，那它是如何实现进程不被结束的呢？看下图 03-17：



选择猎剑——> 扩展功能——> 内核代码扫描就可以得到上图。

当然了，正常情况下，列表应该是空的，但当我运行起 IceSword.exe（冰刃）后，再检查，就会出来上图中的 7 个 Inline-HOOK 项了。

内核代码扫描，是对内核代码中的 Inline-HOOK 进行检查，并列出被 InlineHOOK 的项。

上图中，IceSword.exe 的驱动 IsDrv122.sys，HOOK 了 7 个函数，前六个的功能分别是“结束线程、结束进程、创建进程、创建线程、打开线程、打开进程”后面的那个与进程无关我们不去理它。但从上面 6 个 Inline-HOOK 来看，你应该知道为什么他不会被结束了吧？

还不能确定吗？那么，选中这些 Inline-HOOK 项，然后再在 Inline-HOOK 的列表中按右键，选则“恢复选中的 Inline-HOOK”，然后，你再打开任务管理器，试试结束 IceSword.exe 的进程，是不是可以结束了？

讲到这里，不知朋友是否发现了一问题？我上面说了，NtQuerySystemInformation 是负责查询进程的，但是那个隐藏进程的测试程序 Hide.exe 运行起来后，无论在“SSDT 检查”中还是在“内核代码扫描”中，都没有任何有关这个函数的任何 HOOK 或 Inline-HOOK 的痕迹，这又是怎么回事呢？

不要着急，我们接着讲更深层次的进程隐藏技术。

简单的总结一下：**程序就像是为了实现某一目的而定的计划书；进程呢？就是组织工人分配资源开始执行这份计划；而 Windows 操作系统呢？就是一个为我们管理电脑也是管理这些执行计划的工人的服务管理公司；我们有什么要求呢？就交给服务员将我们的要求提要给 Windows，由 Windows 来组织工人执行我们的要求，并将结果返回给我们。如果木马替换了服务员、更改了 SSDT 表、或替换了 Windows 服务公司某职能部门的人员，我们得到的可能就是一个错的结果，或我们的要求得不到执行，就像结束进程一样。**

如果不替换任何人员，也不更改 SSDT 表，还有没有办法隐藏进程呢？有的，那就是更改系统的数据结构。

举例来说，我们查询进程其实就是向 Windows 发出“查询当前正在有哪几组工人正在工作”的请求，而 Windows 在内部又是如何查询的呢？他当然会去人力部，因为人力部有工人的档案啊，在职的离职的都有，只要将档案一翻，就明白当前有多少工人了，再把结果给我们，它的工作就完成了。

更改数据结构，也就是更改档案了，虽然人员都没问题，但档案已经被修改了，木马会偷偷的潜入人力部把它自己的档案从人力部的档案柜中偷偷的销毁，这样，当系统去查询的时候，就查不到他的档案了，系统会认为没有这组工人，虽然查询过程与查询的人员都没有问题，但结果仍然是错误的。当然了，进入人力部是需要有一定权限的，不是谁都可以进去销毁东西的，注意到与 Hide.exe 在一起还有一个 Sys.sys 文件吗？这个文件也是一个驱动程序，它的工作就是获取 RO 权限，以便顺利的进入内核更改数据。

而 Hide.exe 之所以可以隐藏，就是因为它把系统活动进程链中（即人力部）的进程数据更改了。

那为什么 Windows 查不到它，专业工具却可以查到呢？这个道理很简单，因为 Windows 是正规公司，它的工作流程是固定的，查询工人的事情，他就是去人力部，人力部没有相关工人的信息，他就认为没有这工人，他并不会尝试去其它地方看一看。

其实，工人们是不可能只在一个地方登记的，进入公司要在人力部登记、领工具却还要在仓库登记、发工资要在财务部登记、吃饭要在食堂登记……。

如果木马只是抹去了活动进程链中的数据，那么还可以从窗口、线程、句柄、对像、Csrss.exe 中等许多地方找到它的信息。

当然了，木马也会尽力的把自己的痕迹全部抹去，并且已经有马儿这么做了。

于是，基于线程调度链的进程枚举技术就出现了，什么叫线程调度链呢？在系统中 CPU 的时间分配是以线程为单位的，一个程序要想得到执行，必须要有线程存在于这个线程调度链中，不然，他得不到时间的分配，也就得不到执行的机会。

还以我们上面的例子来说明，木马可以抹去人力部的资料，反正已经进来了，那里有没有也无所谓了；也可以抹去仓库中的资料，工具已经到手了，抹就抹去吧；也可以抹掉财务的资料，豁出去了，为了完成木马大业就不要钱了；但是他不能抹去食堂的资料，不是么？食堂可是按人员档案备饭的，抹掉了，没他的资料也就意味着没他们的饭吃，还不得给饿死啊？而这关键的他无法抹去的资料也就是系统中的“线程调度链”，抹去了，他自己活不下去。

但不要小瞧了木马的智慧，它们也有自己的法子，什么办法呢？他们准备了一份假资料，在有人查询时呢，就用假资料替换掉真资料，而在食堂做饭时呢，再用真资料替换掉假资料，也就是自己构造了一个进程调度链。

开始时还真是没人发现，不过，没有不透风的墙，出来混总是要还的，这技术最后还是让人知道了，即然知道了他们会随时替换，那查询时当然也就知道注意了，于是它又开始想其它办法，斗争就这么持续了下去。

总之，安全与木马的斗争还远远没有结束，胜负还仍然没有分出（好像也很难分出），剿杀木马的工作仍然任重而道远，还我清静河山的愿景，还需我们大家齐心合力的来实现。

上面对“进程”从外到里由浅到深的都讲了讲，当然了，讲的绝不对不全，因为实在太多了，就以隐藏来说，各路高人的各种奇思妙想多不胜数，多到根本无法尽述，比如欺骗眼睛的技术，就是你查时不管你，但你看时却自列表中抹掉，让你看不到，就像明明真实的资料放到了你面前，但偏偏有一个阴影挡住了某处。

但无论如何，邪恶永不能战胜正义，只要我们都关注，总有它们无处藏身的一天。

另外，不得不说的是，查看进程只是检查手段的一种，结束进程也绝不意味着就清除了木马。所以，我们还要继续下去。看看还有什么地方可以抓到它，清除它。

而下一章，就是另一个关键点，自启动项的检查、隐藏与清除。

进程篇到这里就讲完了，接下来是下面的这些：

木马查杀深度剖析之自启动项篇（一）

四、木马的查杀之自启动项篇

自启动项的检查与清除，毫无疑问是查杀木马的关键手段与方法。而且，清掉木马的自启动项，让其自然而亡，是最优雅的杀马方法，不那么暴力也就轻易不会遇到反击。对驱动级的或注入型的木马，这种手段更显其优越性。

而自启动项的选择与设置，更是一种创意的体现，一些非技术型的木马通常可在此看到其作者非同一般的创造性思维。

自启动，顾名思义，就是无须用户干预而自行启动的程序，按启动方式又分为两种，一种是开机自运行的程序；一种是触发式启动的程序。

下面我们将分别来解剖之，但在此之前，我们先要学习一些基础的知识：

1、注册表基础

由于大多数的自启动位置都在注册表中，所以，首先，我们需要了解“注册表”是什么。

注册表从功能上说，它是一个存储各种设置信息的数据仓库，系统的全部设置几乎都存在那里，比如：你用的是什么墙纸、什么屏保、IE 的首页、IE 窗口的大小等等。当然了，开机时需要加载的驱动、开启的服务、运行的程序等等也都存储在这里。

而从实质上来说呢，注册表其实是由一些记录配置信息的文件组成的，这些文件中的大部分存在“\Windows\System32\Config\”目录下，还有一部分存在用户配置文件夹中，也就是下面将要讲到的 03-24 图中的用户文件夹中。

这些文件有一个很难听的名字叫做储巢，也就是朋友可能听过的 HIVE 文件。

由于注册表对系统实在是太重要的了，任何损坏都有可能造成系统彻底的崩溃，所以，系统对注册表的保护也是很严密的，正常情况下，你无法对注册表 HIVE 文件进行任何的直接操作，看下图 03-18：



你不仅无法打开、修改，你甚至无法进行拷贝操作。而系统保护注册表的手段，就是由系统以独占的方式打开注册表文件，这样你的任何针对此文件的操作，都将被拒绝。下面，我们来验证一下，看下图 03-19：

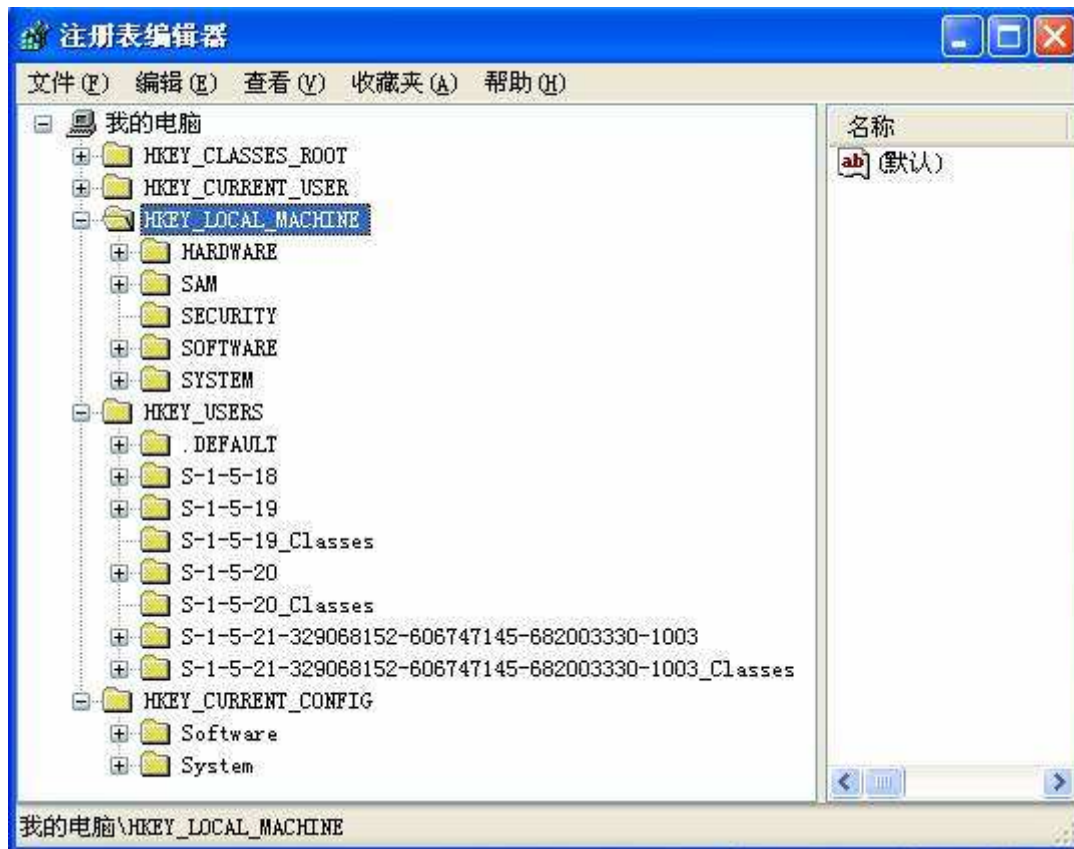


打开狙剑，选择“进程管理”，在进程列表中选“system”进程，再选择“查看打开的文件”，就可以看到系统打开的所有文件，看上图中蓝条选中的那一项，是不是就是我们无法进行操作的“\\Windows\\System32\\Config\\system”文件呢？

注意：狙剑提供了关闭其它进程打开的文件的功能，关闭后，本来无法操作的文件就可以进行正常的操作了。但是，那是对付木马的，而不是对付系统的，因为木马也同样可以利用独占打开的方式来禁止我们对其文件的访问。关闭系统打开的关键文件将发生无可预料的结果，切勿尝试！（关于文件相关的内容，我们在以后的章节中会有详细的讲解。）

那我们是否就无法操作注册表了呢？当然不是，系统在保护注册表文件的同时，也提供了操作注册表的工具-“注册表编辑器”。

在“开始”菜单中，选择“运行”，输入“regedit.exe”就可以打开注册表编辑器，打开后的情况是这样的，看下图 03-20：



上面的各项称为“键”，最上层的是“根键”下面的叫做“子键”，其与上面我们说的注册表文件一一对应，对应关系如下：

```

1HKEY_LOCAL_MACHINE\SAM→Windows\System32\config\sam
1HKEY_LOCAL_MACHINE\SECURITY→Windows\System32\config\security
1HKEY_LOCAL_MACHINE\SOFTWARE→Windows\System32\config\software
1HKEY_LOCAL_MACHINE\SYSTEM→Windows\System32\config\system
1HKEY_USERS\.DEFAULT→Windows\System32\config\Default
1HKEY_USERS\S-1-5-XX_XXX→\DocumentsandSettings\<用户名>\Ntuser.dat
1HKEY_USERS\S-1-5-XX_XXX_Classes→\DocumentsandSettings\<用户名>

```

```
>\LocalSettings\ApplicationData\Microsoft\Windows\Usrclass.dat
```

还有其它的键呢？又对应哪些文件呢？呵，这要从注册表的数据类型说起了，除了上面的键外，其它的几个键是属于 REG_LINK 类型，直翻过来就是“注册表链接”了，这种类型的键是透明的指向另一个键的一个链接，也可以理解成这些键只是另外的某个子键的快速入口或某类相似键的汇总而已，本身并没有文件，其链接的键对应如下：

```

1HKEY_CLASS_ROOT 中的内容来自于下面的两个注册键：HKEY_LOCAL_MACHINE\SOFTWARE\Classes
HKEY_CURRENT_USER\SoftWare\Classes

```


1HKEY_CURRENT_USER 中的内容是当前用户的信息，来自于
HKEY_USERS\S-1-5-XX-XXXX→\DocumentsandSettings\<用户名>\Ntuser.dat

1HKEY_CURRENT_CONFIG 中的内容来自于
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\HardwareProfiles\Current

当你修改有链接关系的任一键中的数据时，对应的链接键也会体现出来，比如：你修改了 HKEY_CLASS_ROOT 中的内容，那么你的修改也同时体现在了 HKEY_LOCAL_MACHINE\SOFTWARE\Classes 中，因为这本就是同一个文件中的同一部分数据。

下面我们再简单的说一说 SID（安全标识符）也就是上面的“S-1-5-XX-XXX”，其后面部分我用“XXX”取代了，后面是不定值。

SID 前面的定义是固定的，比如：S-1-5-18 是本地系统账户（LocalSystem）S-1-5-19 是本地服务账户、S-1-5-20 是网络服务账户等（更多信息，请参考专业资料）

而 SID 后面的部分是全球唯一的，就是：329068152-606747145-682003330-1003 那部分，在注册表中这个是保存了当前用户的配置信息，也就是 HKEY_CURRENT_USER 指向的部分。（关于用户的概念请参考后面的相关章节）

接下来，我们再看看每个键中都存储了哪些信息：

1HKEY_CURRENT_USER 中存储了一些与当前登录用户有关联的数据，也就是你个人的一些设定都存在于这里。

1HKEY_CLASSES_ROOT 中存储的是文件关联和 COM 对像的注册信息，什么样的文件由什么程序来打开就是存在于这里。

1HKEY_LOCAL_MACHINE 中存储的是系统相关的信息，比如：驱动、服务等。

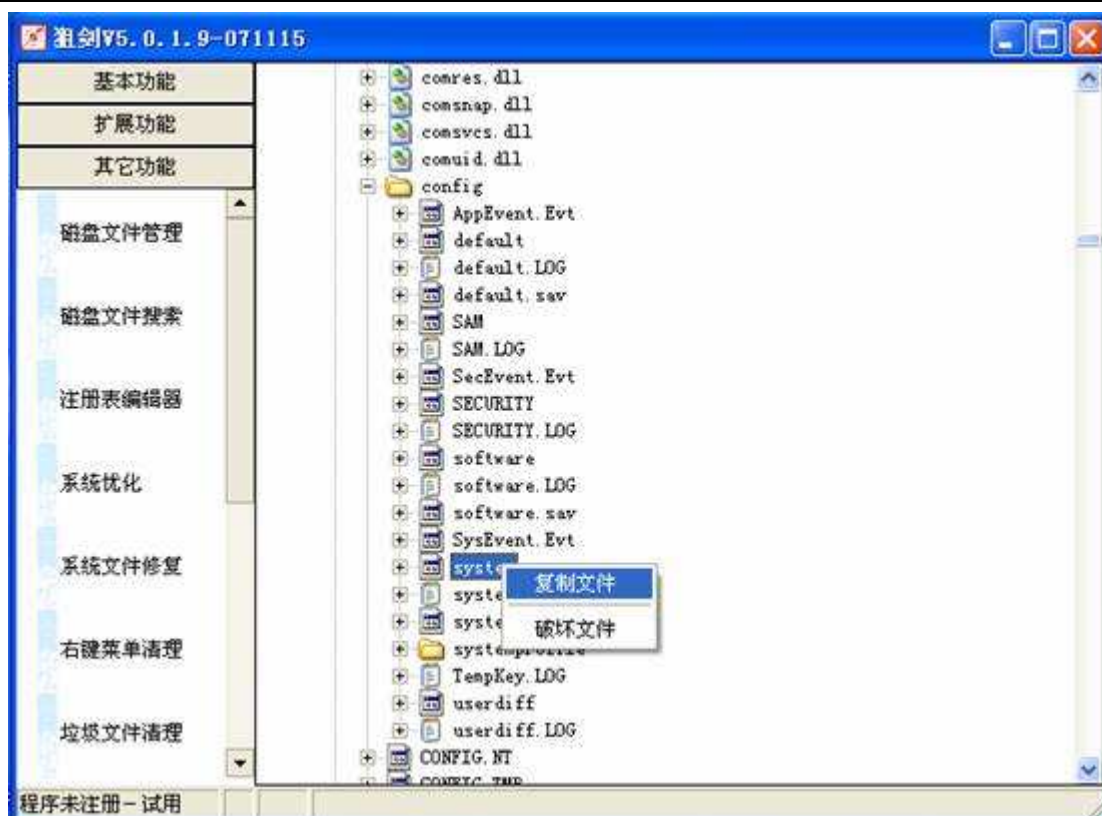
1HKEY_USERS 存储的是所有账户的信息。

1HKEY_CURRENT_CONFIG 中存储的是当前硬件配置信息。

呵，上面都是一些基本的知识，看起来有些枯燥，能理解就理解，理解不了就算了，并不影响后面对木马的查杀。这里之所以讲这些，是提供给想深入了解系统知识的朋友的，要想成为高手就不能不了解这些。

下面我们再来点动手的，提一提精神。那些禁止我们打开、修改甚至不让我们复制的注册表文件我们难道真的就没办法看看里面是什么吗？嘿，不知道有没有好奇心跟我一样重的朋友。想当年，我可是想尽办法也想看看怎么把那些文件给打开的，呵，没办法，好奇心就是重。我们下面就动手来看看，那里面都是些什么神秘的东西，系统居然看都不让我们看。

先看下图 03-21：



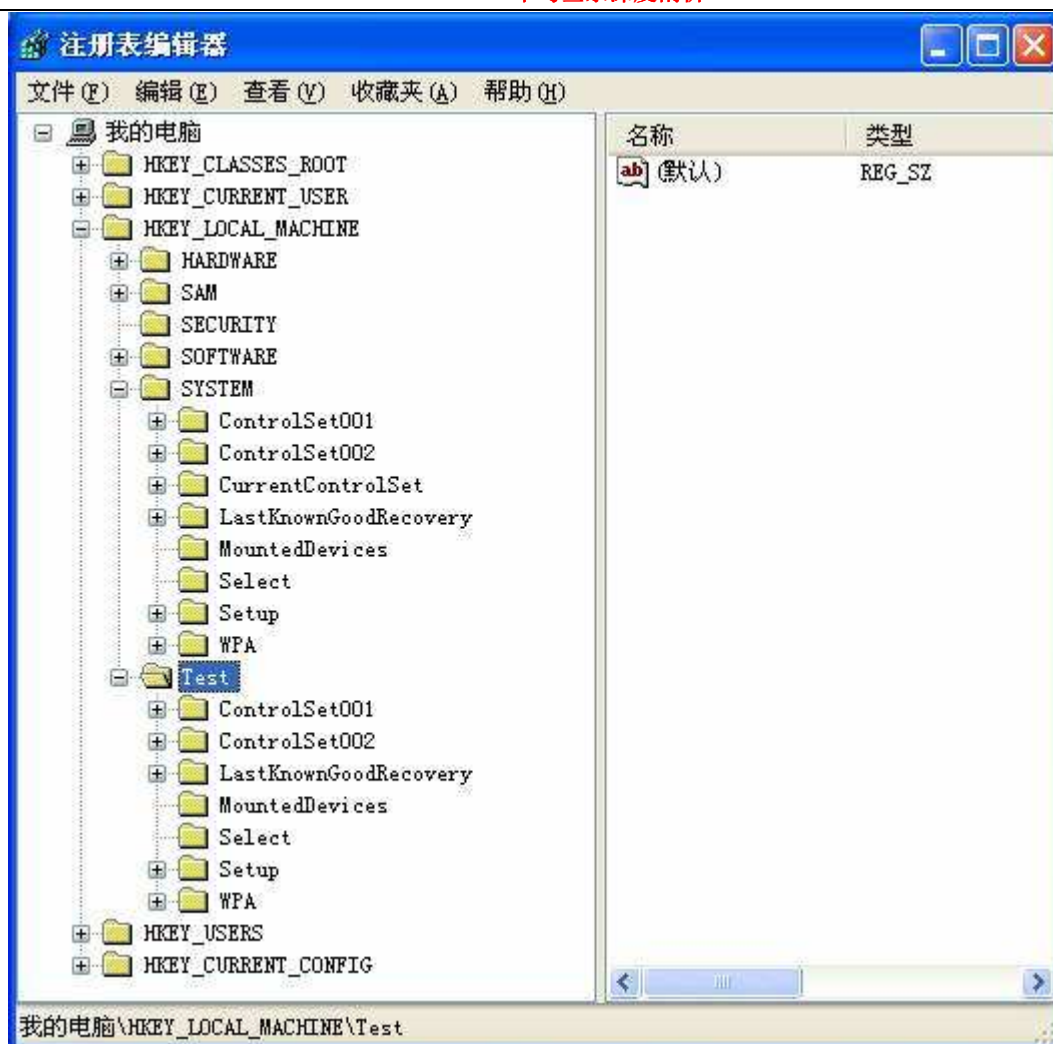
打开狙剑，选择“磁盘文件管理”，在打开的选择框中选择系统盘，然后找到 Windows\System32\config\目录下的注册表文件，按右键，选择“复制文件”。在弹出的保存框中，给新文件起个名字。OK 了，我们成功的复制出了系统本来不想让我们看到的注册表文件。（怎么复制这些不能复制的文件、怎么删除不能删除的文件等我们会在后面的“木马的查杀之文件篇”中统一讲解”）

文件复制出来了，我们怎么来看里面都是什么呢？两种方法，一种是用十六进制编辑器查看，当然了，这种文件是特殊格式的文件，用十六进制编辑器看，看得一定是有些晕的，要做好心理准备哦。第二个方法就是利用 Regedit.exe（注册表编辑器）的“加载配置单元”的功能。

这里，我们讲一讲第二种方法，打开注册表编辑器，先选中 HKEY_LOCAL_MACHINE 键（注意：这一步不要省略，不然加载配置单元项是灰的，也可以选中 USER 键），然后依次选择“文件”→“加载配置单元”。如下图 03-22：



在弹出的选择框中选择我们刚才复制出来的注册表文件，在项名字中随便输入一个名字，这里我输入的是“Test”，然后，就加载成功了，看下图 03-23，是不是多出了一个 Test 键呢？这个键跟 System 键的内容是一样的，因为我们复制的就是 System 注册表文件：



注意一下儿，Test 里面少了这个键“CurrentControlSet”，为什么会少一个呢？不用我说朋友们也应该知道为什么吧？什么，不知道？那再看看他的名字“**Current-Control-Set**”这回知道了吧？对了，这个键也是个链接键，是从 ControlSet001 与 ControlSet002 中链接的内容，是“当前的”设置。所以，文件中是没有的。而注册表编辑器是从内存中读取的注册表内容，因此是有的。

选中“Test”键，在文件菜单中选择“卸载配置单元”就可以卸掉这个键了。

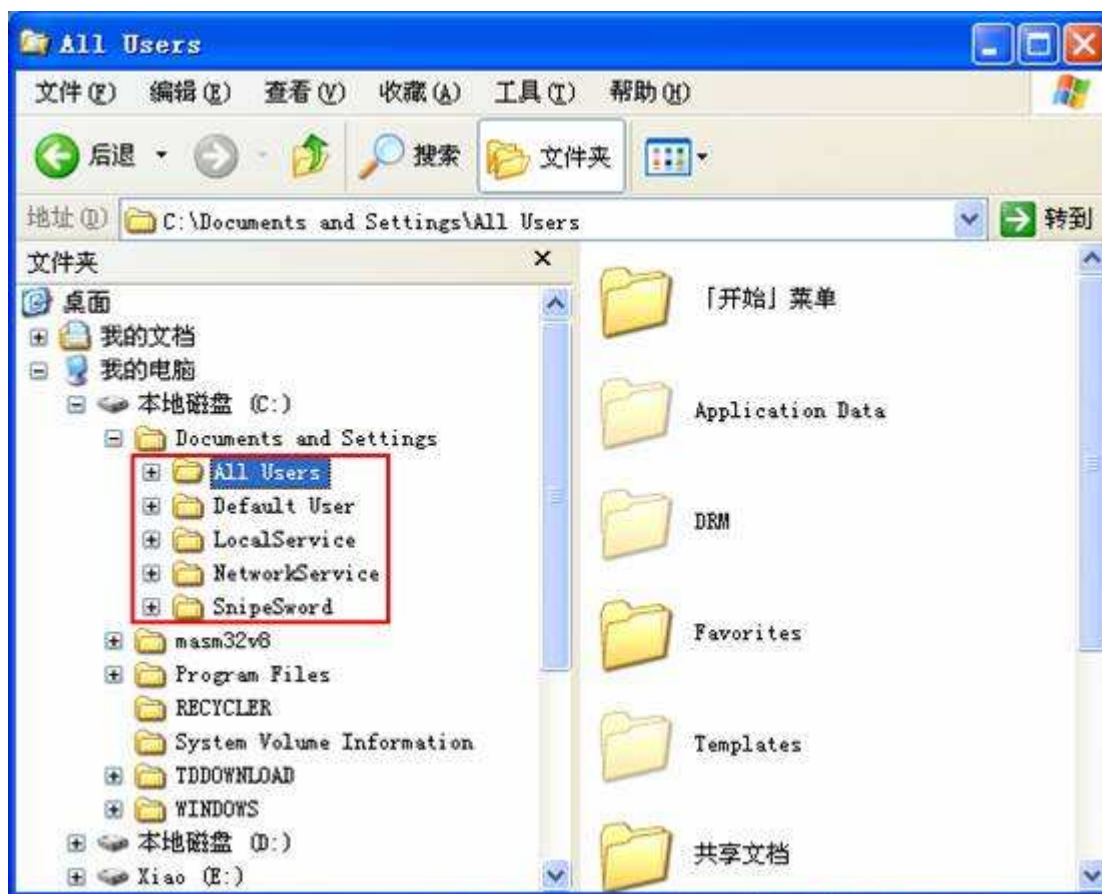
好了，基础知识讲到这里，应该是够用了，我们接下来就分别讲解一下子，开机自动运行的程序与触发式启动的程序。

木马查杀深度剖析之自启动项篇(二)

2、开机自运行的程序

“开机自动运行程序”是操作系统为方便用户的使用而提供的一个方法，让每次开机都须频繁操作的一些工作得到自动的执行。而木马只是利用了这一方法，来完成自己的非法启动目的罢了。

最经典的启动位置，也是操作系统特别提供给用户的位置就是“启动文件夹”，在开始菜单中选择“所有程序”，里面的“启动”就是了。放在此文件夹中的所有程序，都将在下次启动时用户登录后，自动启动起来。这里面涉及到“用户”这个概念，先看下图 03-24:



红框中的就是当前系统中的用户，由上到下分别是“所有用户”、“默认用户”、“本地服务账户”、“网络服务账户”、“SnipeSword 用户（当前用户）”。前面四个是系统创建的，后面的那个是我们自己创建的用户（名字是我们自己取的）。在“所有用户”、“默认用户”与“当前用户”中都有一套大致相同的配置文件，也就是说都有一个“启动文件夹”。

启动文件夹是系统提供给我们使用的，但不知道从什么时候开始，就已经很少有人再用这个位置来启动程序了，包括一些正常的需要自启动的第三程序，也都不再利用这里来启动，而是直接操作注册表来实现程序的自启动。

与启动文件夹相比，注册表操作对用户来说，是更神秘而不易查看与操作，我不知道没有技术原因放弃启动文件夹而利用注册表来启动的那些正常软件是出于什么考虑或出于什么目的，但这种形为，却有必要BS一下儿。

下面我们就接着讲注册表中的自启动项，将那些正常的、或非正常的自启动程序，从我们的机器中揪出来，我们的机器我们作主，即使是正常的程序，大量的开机自运行也会造成系统资源的不必要的浪费与消耗，使系统的运行速度减慢。所以，清理自启动项，并不仅仅是查杀木马，这同时也是优化系统时很重要的一步。

我一直对那些为了商业竞争而漠视用户权力与自由的所谓正当软件深恶痛绝，我绝不想因为一年中仅用了一次某一软件的某一功能，而就任由这软件在 365 天中每天都自动运行起来，消耗掉我宝贵的系统资源。而菜鸟的机器在同样硬件配置的情况下，通常要比高手的机器慢上很多、难用上很多，这些垃圾软件对资源的无谓占用就是很大的一个原因。

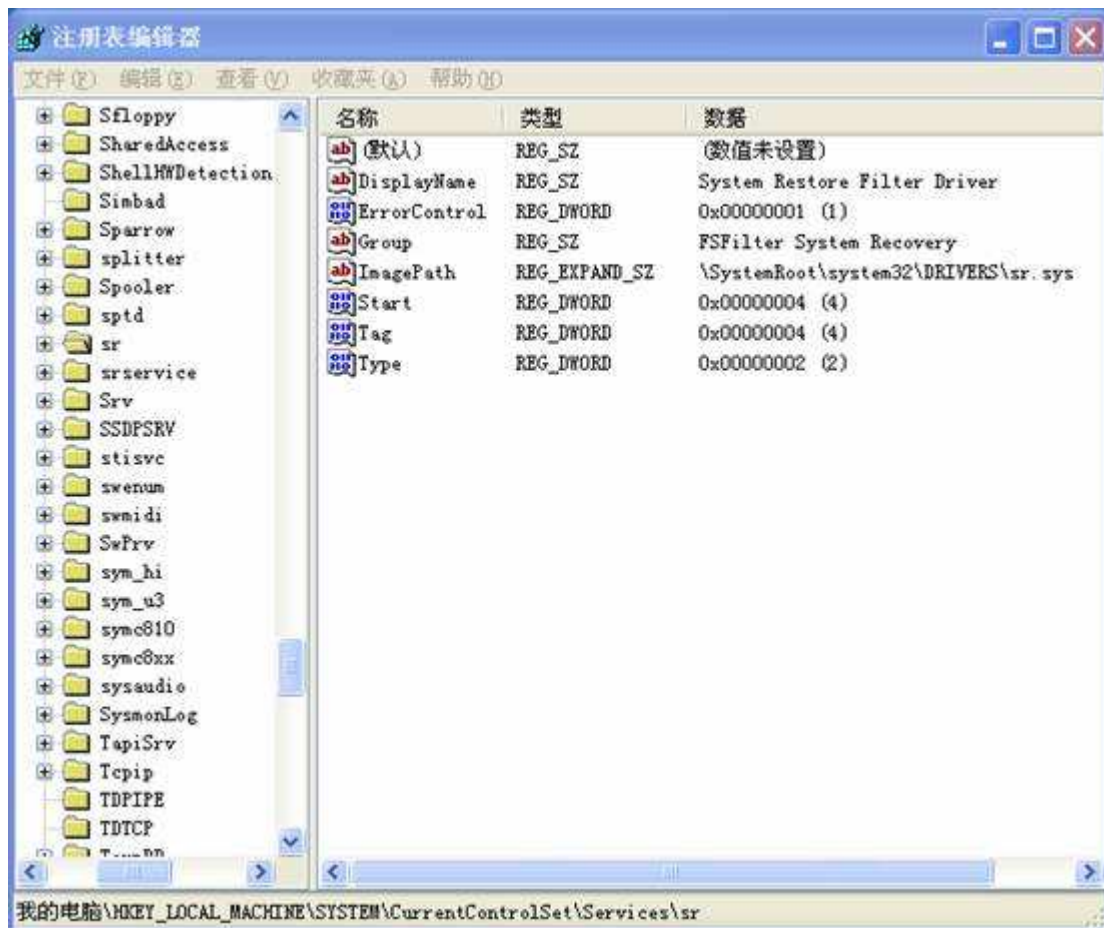
OK，闲话少说，我们下面步入正题。

最早加载启动的是驱动程序，这是可以理解的，因为硬件设备与底层功能是由它们来实现的，不先加载它们哪行呢？比如：在加载文件系统驱动之前，所有对文件的读写都是无法实现的，你想能不先加载这些驱动吗？

系统启动时，由引导程序 Ntldr 来读入 system 注册表文件，加载里面列出的驱动程序。细心的朋友这时可能会有个疑问，文件系统驱动是在这时加载的吗？回答是肯定的“是的”，那按上面的说法，文件系

统加载前，对文件的读取是无法实现的，那么 Ntldr 又如何读取 system 注册表文件的呢？呵，能看到这个问题的，我不得不赞一句真是细心啊。答案是：因为 Ntldr 中内置了只读的 NTFS 与 FAT 文件系统代码，虽然很简单但足够用了。再追问下去，Ntldr 也是以文件形式存在的，那 Ntldr 这个文件又是如何读取的呢？Ntldr 是引导扇区中的引导代码加载的，而引导扇区中有更加简单的文件系统读取代码，区别是越向上越简单，一直到文件系统驱动接手后，才是完整的文件系统代码。引导代码中的文件系统代码简单到只能读取根目录中的文件，所以 Ntldr 只能放在根目录；而 Ntldr 中文件系统代码已经能读取子目录中的文件了，所以驱动是可以放到任意目录的。

我们接着来讲驱动的加载，驱动肯定不是一个，那先加载哪个后加载哪个呢？我们先来看一个图 03-25：



上图中是一个典型的驱动在注册表中的注册内容，系统加载时，就是依照这里的内容进行驱动加载的，这个驱动在注册表中的：HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sr 键下，也就是在 System 注册表文件中。这是 XP-系统还原所需要的一个文件过滤系统驱动。

注册内容中的各项含义如下：

1DisplayName: 这是服务名称，在前面一章中我们讲到服务管理器时讲过，这个名字就是在服务管理器中显示的名称，没什么要紧的，随便起一个就行。

1ErrorControl: 错误控制码，可以取值为，0→ 忽略任何错误；1→ 如果出错，显示一个警告；2→ 如果出错，且有最后一次正确配置则启动最后一次正确配置，否则继续引导过程；3→ 如果出错，且有最后一次正确配置，则使用最后一次正确配置，否则显示蓝屏崩溃。这里的“最后一次正确配置”是指在启动过程中按 F8 键进入高级菜单后的其中的一个选项，利用这个选项通常可以正常进入系统。

1Group: 组的名称，这个根加载顺序就有关系了，我们放到后面详细讲。

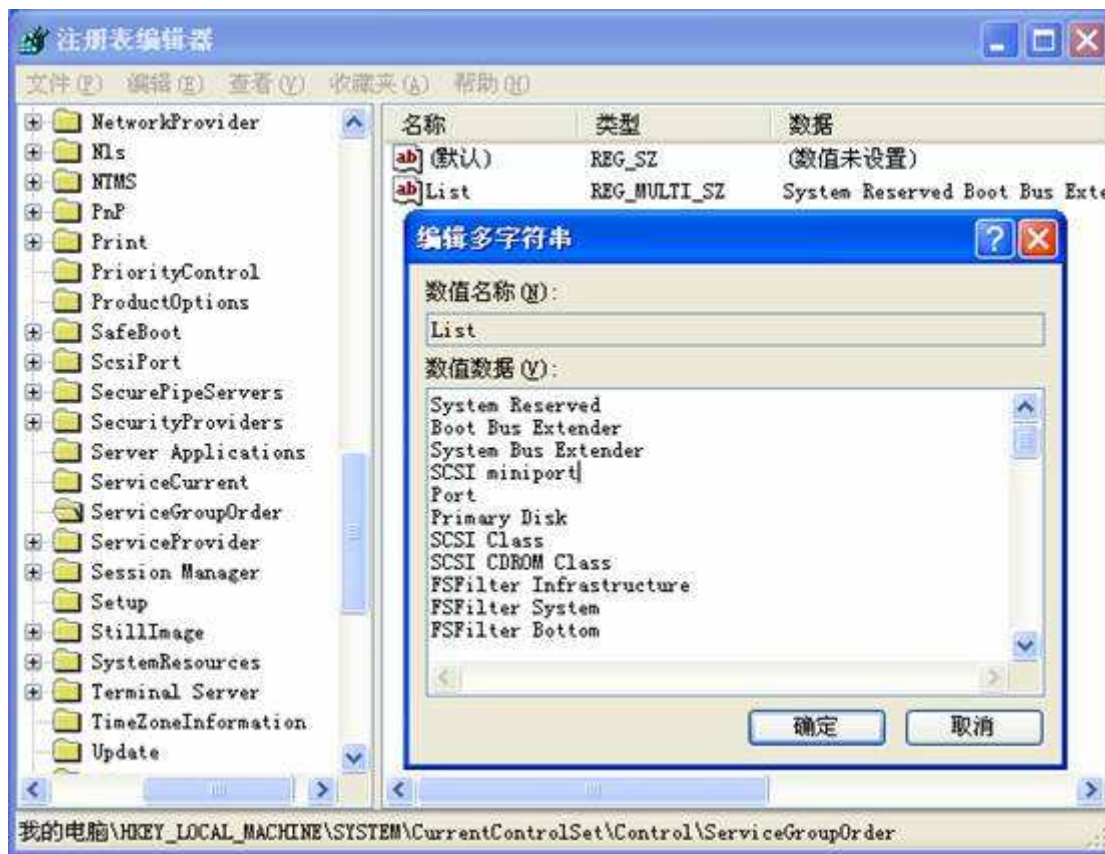
1ImagePath: 驱动或服务的执行文件路径。需要注意的是，如果一个驱动有这一项，那么系统会根据这一项中标明的文件进行加载，如果驱动中没有这一项，系统会自动在 Windows\System32\Drivers\目录下寻找与注册键同名的驱动加载。

1Start: 驱动或服务的加载启动顺序。可取值为, 0→Ntldr 预先加载此值为零的驱动, 在引导过程中这类驱动一直待在内存中; 1→ 在标明为 0 的驱动初始化之后, 值为 1 的驱动开始被加载到内存中并初始化, 其是在内核初始化的过程中加载的 (参见进程篇的系统启动过程)。2→ 在 Services.exe 启动后, 由该进程来加载启动值为 2 的驱动或服务 (Service.exe 的情况请参阅进程篇); 3→Services.exe 根据需要加载这类的驱动或服务, 也就是在服务管理器的启动一项中标明为“手动”的。呵, 这里有个概念性的误区, 标明为手动并不是指一定要由用户来手动启动, 而是由系统识情况启动, 当系统需要相应的驱动或服务提供的功能时, 系统就将自动启动此服务, 而无须用户手动操作; 4→ 驱动或服务并不加载到内存, 当然也不启动了。也就是服务管理器中标为“禁用”的服务, 这一类的服务, 即使系统需要用到其提供的功能, 也不会自动加载。

1Tag: 在组中的顺序, 这也是根加载顺序有关系的, 我们同样放到后面讲。

1Type: 服务的类型, 可取值为, 1→ 设备驱动程序; 2→ 内核模式的文件系统驱动程序; 4→ 已废弃; 8→ 文件系统识别器驱动程序; 16→ 该服务运行在一个只能容纳一个服务的进程中; 32→ 该服务运行在一个可容纳多个服务的进程中; 256→ 允许该服务在控制台显示窗口, 并接收用户输入。

除了上面的 **Start** 决定了加载顺序外, 还有 **Group** 与 **Tag** 来共同作用决定驱动的加载顺序, 下面我们来讲一讲这两个, 先来看图 03-26:



HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\ServiceGroupOrder 键如上图所示, 表明了一个组的概念, 驱动是分组并按组来分别按顺序加载的, Group 决定了驱动是在哪一个组中, 而 Tag 却决定了在同一个组的哪一个位置上。注意一下儿就会发现, Start 值为 0 的驱动基本上都是分到了*前的几个组中。

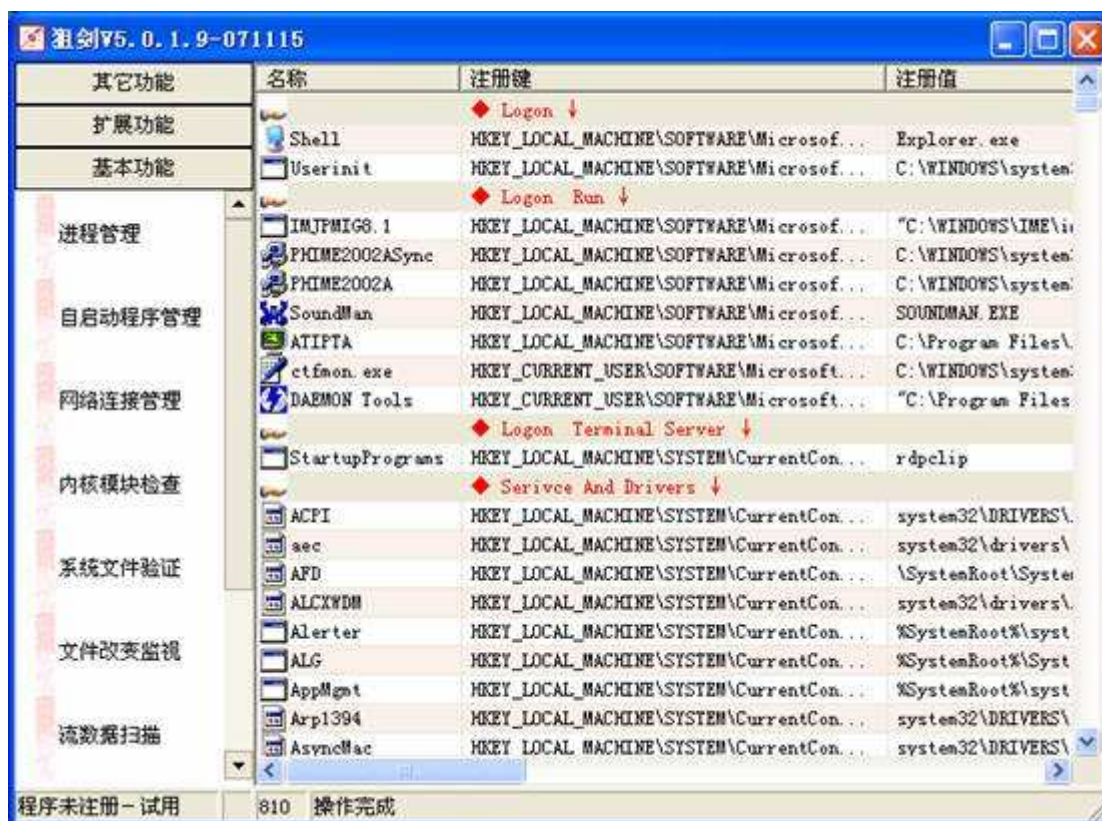
了解了上面的知识后, 只要你有足够的耐心, 那么你就可以将系统中的驱动加载顺序搞出来了。

驱动加载后, 还有一些系统启动过程中需要的程序会得到加载执行的机会, 可以参考进程篇中的系统引导过程来看, 这里就不多说了, 而且应用程序的启动顺序永远无法超越高启动级别的驱动, 所以讲他们启动顺序也没多大意义。

在注册表中可以启动程序的地方很多, 不可能一个个的都讲一讲, 也没那个必要。同样的, 我们也不可能手动来从注册表中查找每一个启动项, 启动项的查找只有依*专业工具来进行, 才是最佳的选择。

启动项管理软件中最出名的无疑是 Sysinternals 出品的 AutoRuns.exe 这是国外的一个专业启动项管理工具，很出名。但是如果用于清除木马，这个工具很显然是远远不够的，就像它的名字一样，这是一个“启动项管理”工具，它虽然列出了很全的自启动项，但却缺少反黑工具所特有的查找被隐藏的启动项、清除被保护的启动项的能力，我相信我这个评价还是很客观的。在后面的“自启动项的保护与清除”一单中我会再详细讲解。

现在我们再看看狙剑的自启动项管理，狙剑毫无疑问是借鉴了 AutoRuns.exe 的启动项，另外加入了一些 AutoRuns.exe 并没有列出的启动项，从全面上来说应该是更胜一筹了，呵，说来也惭愧，站在巨人的肩上总会比巨人高一头的。当然了，最关键的还是狙剑做为一个专业的安全工具来对自启动项进行管理，充分考虑到了木马病毒对自启动项的隐藏与保护，加入了隐藏 HOOK 的恢复，所以，更适合安全相关的自启动项管理，我们看看下图 03-27：



注意，下面标明了自启动项的数量是 810，只是这个数包括了分类条目，所以实际的会少一些，但怎么也应该在 790+，这种数量的启动项，用手工来找，很显然是不现实的。另外需要注意的是，在驱动与服务的枚举中，狙剑将启动值为 4 也就是标为禁用的驱动与服务同样列了出来，这是为了防止有些恶意驱动，在加载完成后，就将自己改为禁用，等关机时再改回 0 值优先启动而考虑的。

对于这大量自启动项的判断上，我们也将用进程判断的方法来进行判断，也就是数字签名验证，在启动列表中按右键选择“隐藏微软签名项”后，就会将全部有微软签名的启动项隐藏，剩下的也就没多少了。

在正常情况下，清除掉所有的非微软签名启动项，是不会影响系统正常使用的。但有些情况是需要注意的，比如：Tcpip.sys 是网络驱动，正常情况下是可以通过微软数字签名验证的。但有些下载软件比如迅雷会修改此驱动以达到最大的连接数，而由于 Tcpip.sys 被改动后，将不再能够通过微软的数字签名验证，所以，在不特别注意下，清除所有非微软认证的自启动项时，就会将这个驱动清掉，导致无法上网。

另外，向进程篇中提到的 Userinit.exe 的情况也是类似的，Userinit.exe 同样为系统必须的程序，但如果这种程序被感染了，那么，直接清掉就会存在问题，当然了，安全程序在清这类启动必须的程序时是不会删掉启动项的，但作为我们用户来说一定要搞清楚哪些是木马的启动项，那些是被修改或感染的系统启动项，对这种被修改或感染的系统文件的处理，最佳方式无疑就是利用“系统文件修复”功能了。而千万不要直接手动清除相应的启动项，清掉后，将导致系统功能出现问题甚至不能进入系统。

所以，对启动项进行清理时，在隐藏微软验证的文件后，仍然要对剩余的进行判断，有针对性的清理，当然，判断的过程是需要经验的，但是，想成为一个高手，这份经验的积累是必须的，世界上没有白吃的午餐，不想做一个使用傻瓜式软件的傻瓜，那么一些必要的学习与充电也就是必须的。第一次清理时不知道，上网查询或求助，第二次就知道了。相信，不长时间就可以完成经验的积累。

最后需要说明的是，新的启动位置仍然在不断的挖掘中，没有哪个软件敢说全，只能是比较全而已，仍然需要不断的补充与完善。

木马查杀深度剖析之自启动项篇(三)

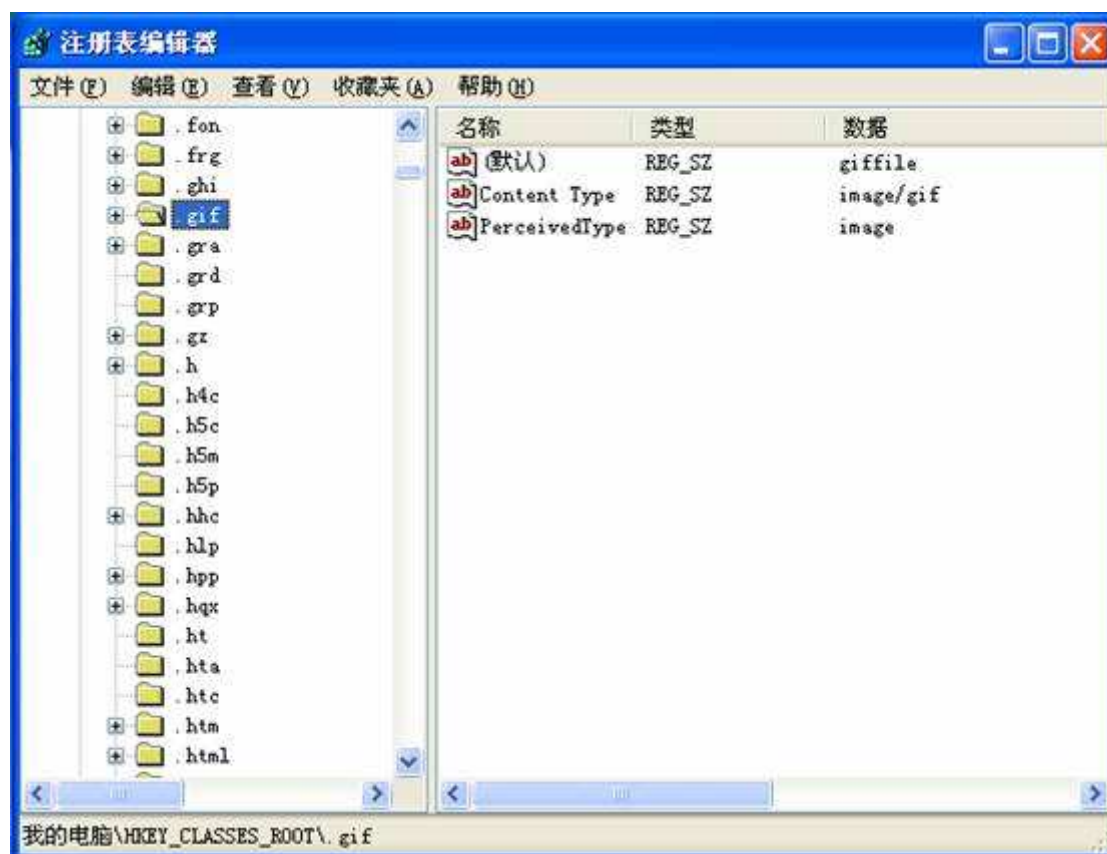
3、触发式启动的程序

什么叫做触发式启动的程序呢？触发式启动程序是指需要用户进行某一操作来触发而启动的程序。

A、最常见的就是文件关联式的触发启动。

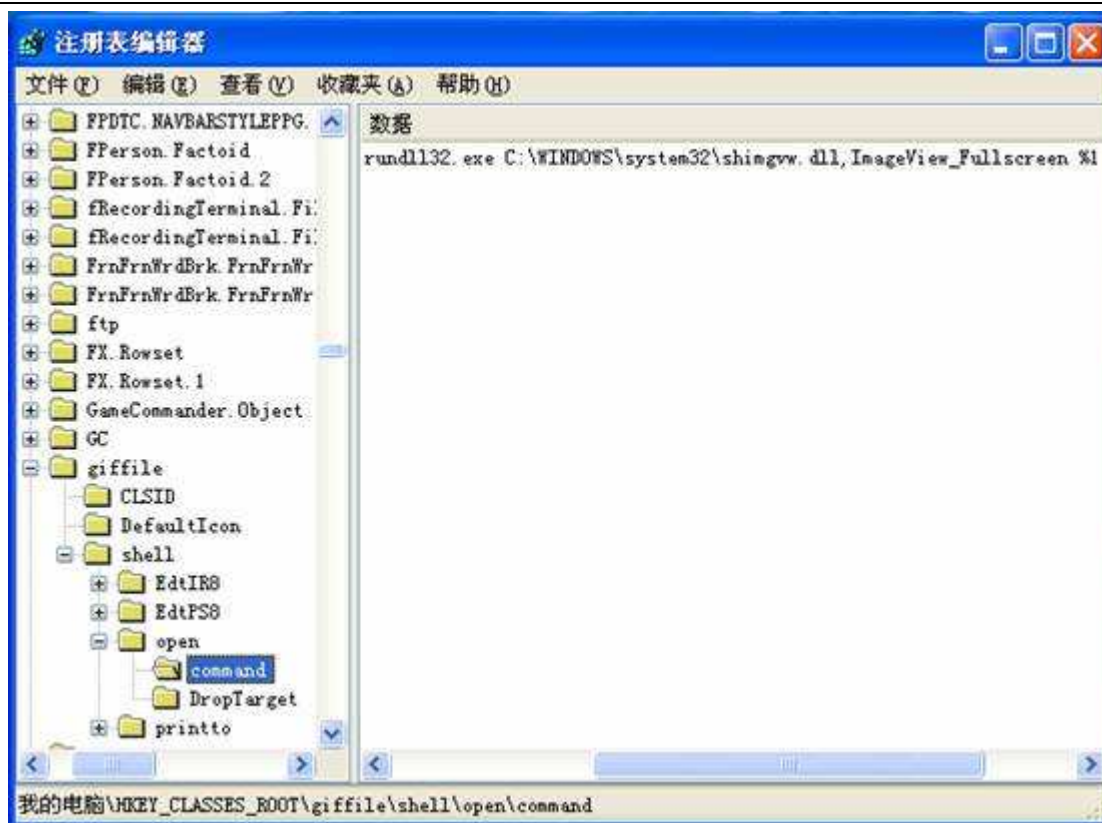
比如说修改 exe 文件关联的木马，每当你运行 exe 文件时都将触发木马程序的执行，同样的，如果修改了 gif 文件关联，那么每当你打开 gif 文件时，就会触发木马程序的执行。

下面我们就详细讲解一下修改文件关联的木马，我们以 gif 文件为例，先看下图 03-28：



如果在“注册基础知识”一章中看仔细了，那么应该知道 HKEY_CLASSES_ROOT 键下存储的正是文件关联信息。看上图，各子键大都是以“.”开始的后面跟三个字母，想一想跟什么比较为像？呵，当然是文件的扩展名了。Explorer.exe、Regedit.exe……后面的扩展名说明了文件的类型，.exe 是 Windows 可执行文件。A.gif、B.gif……后面的扩展名 GIF 说明了文件是 GIF 图片，那么我们如何查看哪一类型的文件是由哪个程序来打开的呢？

看上图中的.gif 键，其“默认”值是“giffile”。然后，我们再在 HKEY_CLASSES_ROOT 键下找到“giffile”子键，如下图 03-29：



在 giffile 下的 shell\open\command 键的默认值，标明了当你双击 gif 文件时，系统所要做的动作，这里的数据是“rundll32.exe C:\WINDOWS\system32\shimgvw.dll, ImageView_Fullscreen %1”，也就是说当你打开 GIF 图片时，系统会执行上面的命令，解释如下：运行 rundll32.exe 来调用 C:\WINDOWS\system32\shimgvw.dll 这个动态库，而参数就是 ImageView_Fullscreen，后面的 %1 在执行中会用你双击的图片名字替换掉。

结果就是打开了图片浏览器来浏览这个图片，我们下面修改一下这个值，改成：“D:\Adobe\PhotoshopCS\Photoshop.exe”%1”那么再次双击 GIF 图片文件时，执行的结果就是运行 Photoshop.exe 来编辑这个图片了，而不是再用图片浏览器浏览。（当然了，D:\Adobe\PhotoshopCS\Photoshop.exe 是我机器上 Photoshop.exe 安装的路径，你需要改成你机器上的路径，其实你也可以改成任何一个程序）

如果木马修改了这个值呢？改成了“C:\木马.exe%1”那么，一旦你双击 GIF 图片文件想浏览一下图片时，系统就会很忠实的依照注册表中的这个值，启动“C:\木马.exe”，从而将木马触发启动，而木马也会在自己启动后，再度调用原有的程序将图片打开，这样，对你来说，是没有任何感觉的，你只知道双击一个图片后，图片打开了，而不会知道发生在这中间的一切。

当前常见的木马对文件关联的修改都是以 exe、com、txt 等常见类型居多，而大多数安全软件也都对这几个常见的关联项进行了检查，但是木马开发者不是傻瓜应该是毫无疑问的事实，明知道常见类型会被安全软件监管后，他们还会修改常见的类型吗？呵，系统中可供修改的文件关联简直太多了。Rar 文件，是常见的压缩格式文件，从网上下载的文件有很多是以这种格式压缩的，修改了它，一旦你打开下载的压缩包时就将启动木马。BMP、GIF、JPEG 等图片文件、AVI、RM 等电影文件，修改任何一个，都将有很大机会被触发，而修改多个呢？

猎剑提供了对所有文件关联进行检查的功能，看下图 03-30：

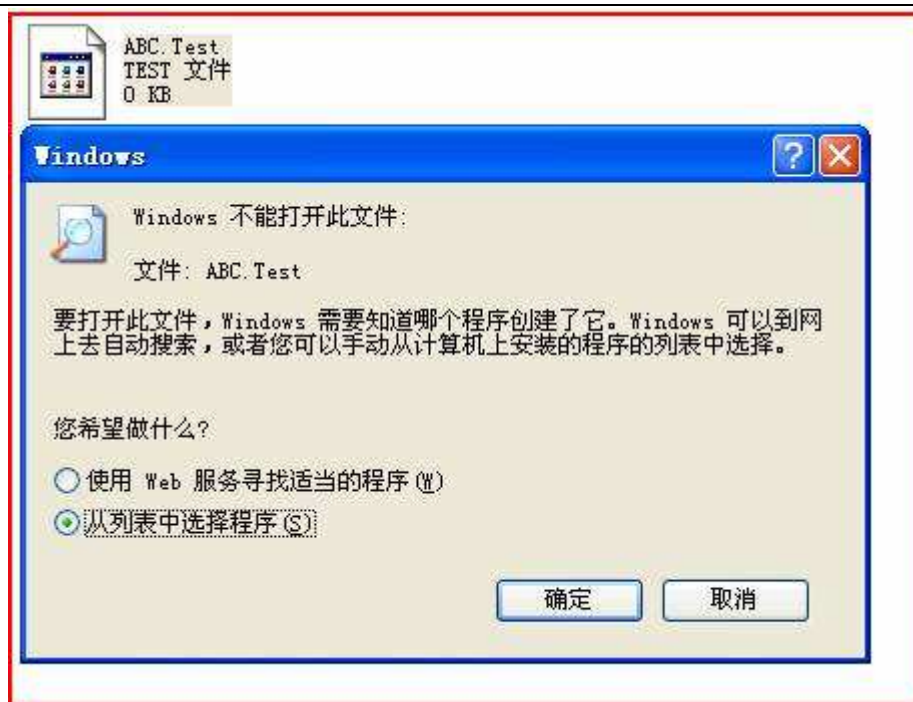


打开组剑，在基本功能中选择“注册项扫描”，就可以对所有的文件关联及 COM 注册项等进行扫描。

查找规则是列出所有非系统程序的文件关联，也就是说如果某一类型的文件并不是由系统程序打开的，那么就会被列出。虽然已经隐藏了上千个系统文件的文件关联，但剩下的仍然会有很多。不过，虽然不少，可是并不难判断，一个程序通常对应着很多种类型的文件，比如，一个暴风影音就对应叫几十种文件类型，很容易就可以判断出这是正常的关联。

当然了，这功能平时用的机会并不多，但是一旦一只马儿清掉后，过几天就死灰复燃时，检查文件关联也许会有意外的收获。

下面呢，我们再说一说文件关联的设置，如果一个文件关联被木马修改了，我们如何改回正常的呢？通过上面的讲解，也许有基础的朋友已经可以通过修改注册表来实现文件关联的修改了。但是我们还有更简单的方法来进行文件关联的修改与设置，看下图 03-31：



当打开一个未注册的文件时，或我们在清掉了一个原有的文件关联后，如上图所示，系统会弹出提示一个窗口。

ABC.Test 的扩展名是“.Test”这是我自己随便取的一个名字，系统当然不知道，对于不知道类型的文件，文件的图标通常就是上面图中的样子，一个标准的文件图标。

这个类型的文件网上也不会有，所以也不用尝试在网上找了，不过，如果一般的可以尝试在网上找一找。这里我们自己来选择相应的文件关联程序，所以，我们选了上图中下面的一个选项“从列表中选择程序”。这个选项的意思是，由我们自己来选择用哪一个程序来打开扩展名是“.Test”的文件。

单击确定后，会出现下图所示的窗口，看图 03-32：



Windows 列了当前系统中安装的所有程序来供我们选择，这里我们选中了记事本，然后勾选下面的“始终使用选择的程序打开这种文件”，然后，点确定，就完成了 .Test 这种文件的文件关联设置。如果列表中没有我们需要的程序，还可以点击“浏览”自己找需要的程序。

设置完成后，以后一旦我们双击扩展名为 Test 的文件，系统就会自动的调用记事本来打开这个文件。

说完文件关联的，我们再接着说下一种。

B、自动播放式的触发启动。

自动播放本是用于光盘的，当插入一个电影光盘到光驱时，系统会自动播放里面的内容，这就是自动播放的本意，播放什么是由光盘中的 AutoRun. inf 文件指定的。此文件的内容，通常如下：

```
[autorun]
open=AutoRun.exe
icon=AutoRun.ico
```

Open 那一行，指出了自动播放时系统自动运行的程序，icon 指出了所显示的图标。后来有人用于了硬盘与 U 盘，在 U 盘或硬盘的分区，创建 Autorun. inf 文件，并在 Open 中指定木马程序，这样，当你打开硬盘分区或 U 盘时，就会触发木马程序的运行。

这类启动，大部分安全工具都进行了监管，在狙剑的自启动项中会列出每个磁盘或分区的自动播放式的自启动程序。

C、感染式的触发启动

这就涉及到病毒了，电脑病毒通过感染正常的程序来实现自己的传播与启动。

计算机病毒是什么呢？其实也是一段程序，只是其完成的是特殊的工作。上面我们说过了，程序就是一份计划书，里面存的是指令序列，标明了程序所完成的工作流程。那么病毒程序的工作流程是什么样子的

也就是说，病毒将自己的病毒体附加到了正常的程序上面，并修改了程序的入口地址为病毒体的执行地址，最后再由病毒体执行完毕后跳回原程序的入口地址来执行原程序的功能，这一切对用户来说都是不可见的。

病毒的清除并不复杂，不外乎就是一个分析，从被感染程序中找到病毒体，清掉，再恢复原程序的入口地址，就 OK 了。但难就难在病毒与病毒是各不相同的，而新病毒又层出不穷，这就使得这种分析与清除工作需要大量的人力与精力。所以，也只有有实力的大公司才有可能很好的维护一个病毒库，并不断的分析新病毒来更新病毒库。

修改式的触发启动，主要是指修改原本为正常启动的环节，实现自身的启动，这个与感染式的比较像，但又不同。他们的修改并非大量的，而是有针对性的。

这种方式隐蔽性很强，但却很难逃过以数字签名验证为主要手段的启动项检查，因为一旦系统文件被修改，即不再能通过数字签名验证，前面提到的修改 Userinit.exe 的木马就是这样的一个例子。

狙剑提供了系统文件修复的功能，其实这一功能也是调用的系统本身自带的系统文件修复，只是进行了少许的加工罢了。比如：有的光盘系统安装目录与原版系统盘并不相同，这样，用系统自带的系统文件修复功能时，即使你插入了系统安装盘，也会提示你找不到系统盘，而改进后的就可以手选择安装目录了。再比如，装机器时，有些装机员习惯在机器上拷贝一份安装目录，这样就不用频繁的插入系统光盘了。而这个硬盘上的目录，通常是不能用于系统文件修复的。改进后的多了一个全盘扫描系统安装目录的功能，你可以指定或让狙剑自动扫描安装目录后，再进行修复。

呵，分类是我自己分的，名字也是自己取的，是否贴切很难说，大家就凑合着看吧。事件是指当你进行某项工作时，比如：下载

用过迅雷的朋友可能有体会，一旦下载东西，迅雷就会自动打开，而无论先前迅雷是否已经运行。试想，如果木马也有同样的本事，我们一下载东西就打开木马，那岂不是很可怕？

- 40 -

总之，启动的方式多种多样，手法也是各有不同，想一网打尽几乎是不可能的，只能是尽力的发现、加入、再有发现、再加入。而以上说的也只是常见的，还有一些并不常见的，我也就不多说了，因为没发现有利用的，这里也就不提了，以免被坏人利用。

下面我们开始聊聊自启动项的隐藏与保护，虽然分成了隐藏与保护，但两者使用的也大多是相同的技术，也就合在一起聊了，能保护也就能隐藏，反之亦然。

木马查杀深度剖析之自启动项篇(四)

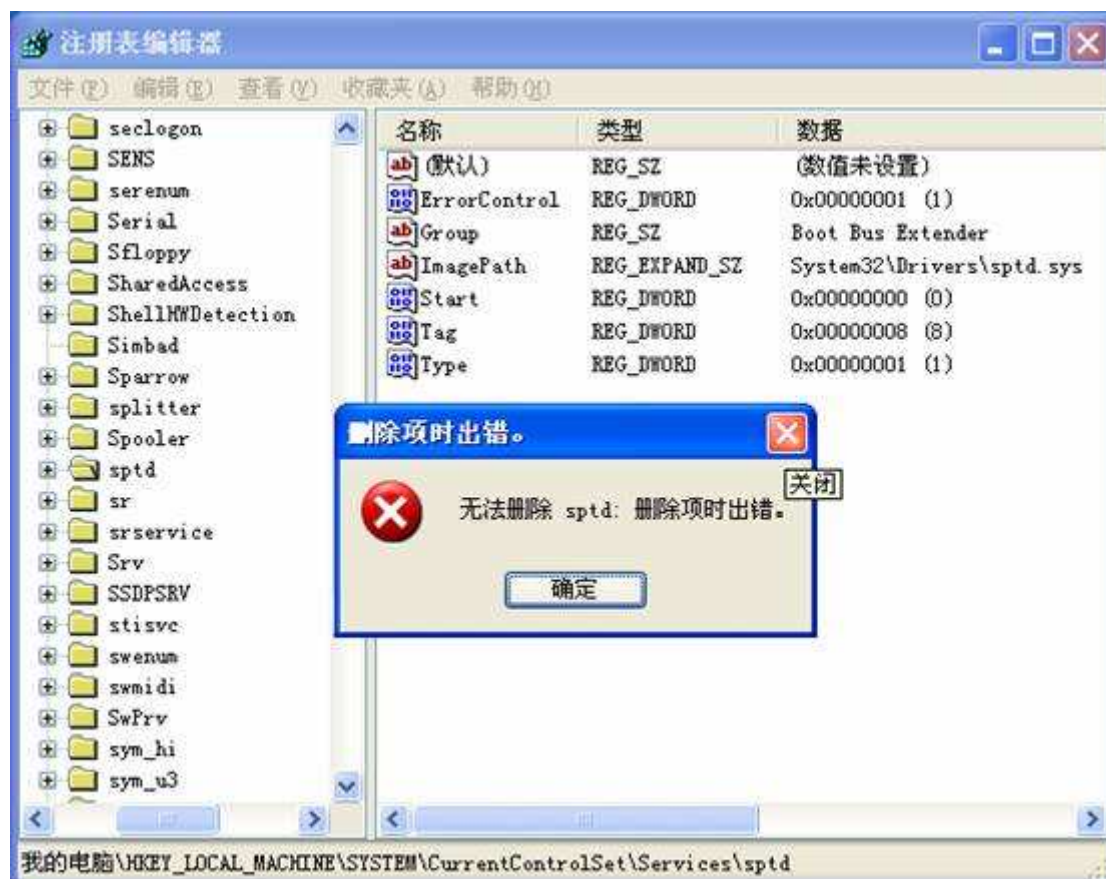
4、自启动项的隐藏、保护与查杀

自启动项的隐藏与保护，跟进程的差不多，不外乎还是 HOOK，系统提供给程序开发人员对注册表操作的功能函数常用的有如下几个：RegCreateKey、RegOpenKey 用于打开创建注册键；RegDeleteKey、RegDeleteValue 用于删除注册键及注册值；RegEnumKey、RegEnumValue 用于枚举注册键及注册值；还有 RegQueryValue 来获取值。另外还有一套*Ex 函数，其实功能都是一样的。

而系统查看或删除注册键与值时，也是用的上面的几个 API。而上面的一些个 API 是用于应用层的，它们又会调用系统内核中的 Nt*系统的功能服务来进行接下来的处理(*代表了 Nt 开头的那些与注册表相关的系统服务)，而 Nt*系统服务又调用了 Cm*底层功能代码来进行操作，当然了，最终的操作都要实现在 HIVE 文件上面去。

进程篇中的 SSDT-HOOK、SSDT-INLINE-HOOK 在注册表 HOOK 时是完全有效的，HOOK 应用层的 RegCreateKey、RegOpenKey 或 RegDeleteKey 都可以实现注册键的防删除，而 HOOKRegEnumKey 就可以实现注册键的隐藏，这种 HOOK 可以对付系统与大多数应用层的安全工具的检查；而 HOOKNtCreateKey 等则同样可以实现隐藏与防删除且层次更深，而 HOOKCm*系统注册表操作函数，则更加邪恶，且已经有这类程序出现。总之，在注册表操作的任何一个环节进行 HOOK，都可以实现隐藏与防删除的目的。

我们再来看看图 03-33：



在安装了著名的虚拟光驱程序 DaemonTools 后，此工具的驱动会禁止用户删除其注册项，SPTD.sys 就是它的驱动。再看下图 03-34：



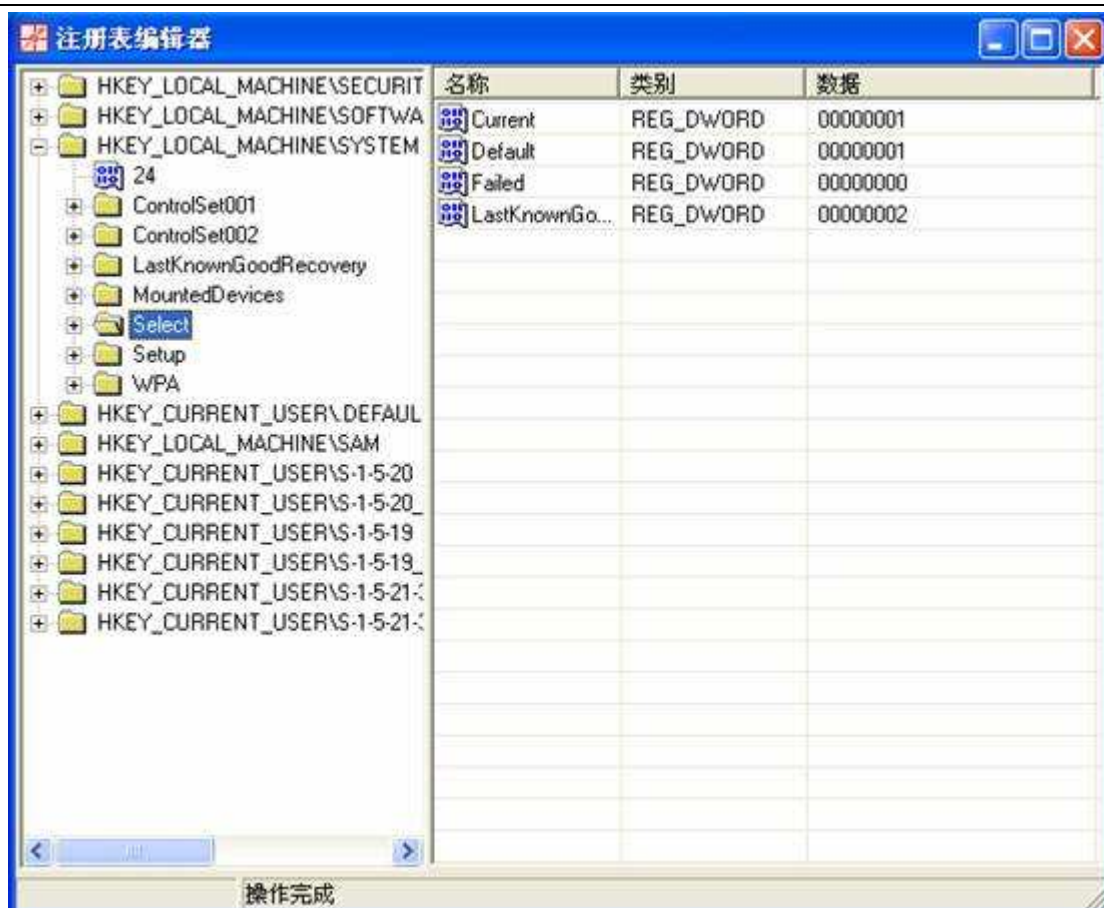
打开猎剑，选择“扩展功能”→“SSDT 检查”就可以看到 sptd.sys 对注册表相关函数的 HOOK。

有了进程篇的相关例子，这里就不再细说了，原理都是一样的，还有对 Inline-HOOK 的检测可以用“内核代码扫描”。内核代码扫描不仅能找到 Inline-HOOKNt*系列函数木马，同样能找到 Inline-HOOKCm*系列函数的木马。只不过 HOOKCm*系统函数的木马层次更深一些而已。

而如果直接用猎剑的自启动项管理对自启动项进行操作，则无须手动检查与恢复 HOOK，猎剑在扫描与清除自启动项时，会自动恢复相关的 HOOK（有些 HOOK 有防恢复机制，也很难用手工来恢复，所以也就没必要非手工恢复它，交给工具去做就可以了）。

注册表由于其特殊性，使得我们多了一个检查与清除木马自启动项的终极手段，那就是直接操作 HIVE 文件，当然了，HIVE 文件的重要性使得系统对其保护很严密，这对直接操作 HIVE 文件造成了一定的麻烦，而风险性也进一步的提高。这使得当前绝大多数安全软件，对操作 HIVE 文件来查杀木马都望而却步，但其效果却无疑是目前最好的。

我们先看看下图 03-35：

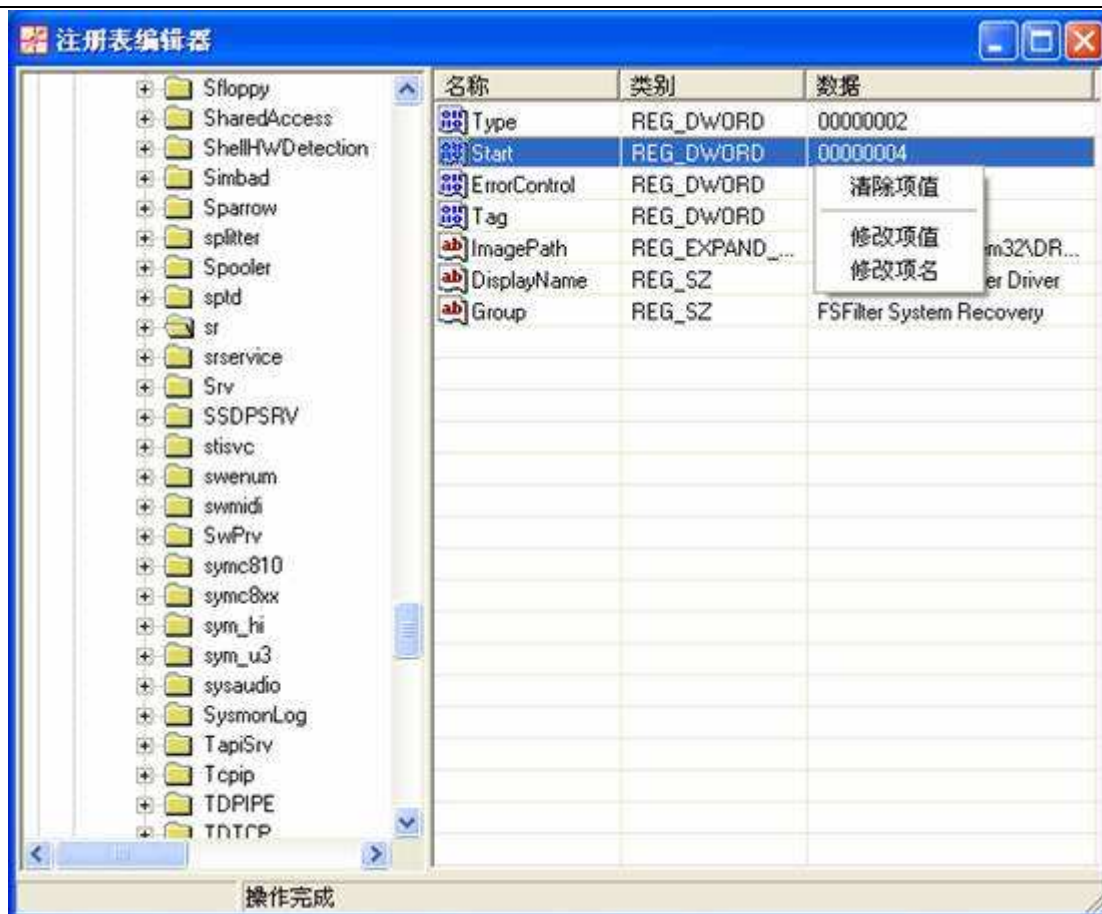


打开狙剑，选择“其它功能”→“注册表编辑器”就可以打开如上图所示的狙剑注册表编辑器，这个编辑器跟系统自带的那个 Regedit.exe 最大的不同就是，狙剑注册表编辑器是直接操作的 HIVE 文件，而没有使用任何的 Reg*、Nt*、Cm*等系统的注册表操作函数，这样，所有的注册表相关 HOOK 对这里的操作将完全没有任何效果。

注意看上图的：**HKEY_LOCAL_MACHINE\SYSTEM** 键，再与我们上面讲到的，复制 System 注册表文件后，用 Regedit.exe 的加载配置单元，加载后的内容是不是一样的？都缺少 CurrentControlSet 这一个子键。

在左侧的列表中按右键，可以选择“仅显示自启动相关项”，这时与自启动无关的项将被隐藏，方便使用注册表编辑器进行木马的查找。而如何使用注册表编辑器在 HIVE 级别上清除木马的自启动项呢？

我们看下图 03-36：



仍然以我们上面用到的系统还原的驱动为例来说明，找到这个键：

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sr

注意在右侧，是这个驱动的注册内容，在上面按右键，可弹出操作菜单，里面提供了几个功能“清除项值”、“修改项值”、“修改项名”。

1 清除项值：这个功能用于一般的自启动项，结果是会将“数据”那一栏的内容清零。比如我们可以将 ImagePath 一项的数据清零，那么对于一般的驱动来说这个操作就可以禁止驱动的加载了。

1 修改项值：这个功能是修改“数据”那一栏的内容，这个用来做什么呢？呵，充分发挥想像，可以做很多事情。比如：将 Start 那一项的值改为 4！为什么改为 4 呢？如果上面注册表基础知识那一章看得仔细就应该知道，改为 4 是禁用此驱动的意思。或将 ImagePath 一项的数据中原来的 Sr.sys 改为：SS.sys 只改了一个字母，但驱动的加载是铁定会失败的，因为系统会在加载时努力的寻找 SS.sys，结果当然是找不到了。

1 修改项名：这个修改的是第一列的项名字，也就是“Start、ImagePath……”等。呵，如何用，我就不多说了，相信也不用我多说了。项名字都是固定了，稍加改动就会使得系统认为这是无效项。

操作 HIVE 无疑是最底层的也是功能强大的，但用起来肯定不如使用自启动项管理方便，所以，一般情况下，这个是用不到的。

但一旦遇到变态的木马时，有这个功能，会让我们感到心里有底。比如：一旦清除木马的启动项，系统马上就蓝屏的情况我就遇到过，这是木马变态般的自我保护，而其保护的基理就是注册了注册表改变通知的消息，一旦其注册表项被改变，他就会接到系统的通知，而他一旦接到通知，就会马上使系统崩溃，让我们对其所做的修改无效。

为什么马上系统崩溃就会使修改无效呢？这是因为系统的缓存机制在起作用，正常情况下，我们的修改只体现在了缓存中，系统通常是 5 秒钟为间隔的将缓存中的数据向磁盘中更新，而修改后马上崩溃，将使得系统来不及向磁盘中写入修改后的信息，导致修改无效。（后面删除文件一章中我们还会讲到这个。）

对这个变态的木马，我的处理，只是简单的将其“Start”中的值改为了4，重启系统后，它就自然死亡了。原理也很简单，直接操作HIVE文件绕过了系统所有的注册表操作环节，系统也不知道我们做了更改，即然系统都不知道，木马自然也得不到通知了。

总之，越向底层走，功能越强大，但操作就越不方便，风险也就越大。像狙剑的“终极修复”就是一个例子。终极修复其实也是一个操作注册表文件的例子，在系统初装时，系统会在Windows\Repair目录下，保存一份注册表HIVE文件的完整备份，这是最原始的也是最简洁干净的注册表文件。

终极修复利用的就是这个备份的注册表文件，用这一套文件取代现有文件，将会使得系统的全部配置回复到初装系统时的状态，那些木马病毒的自启动项自然也就不存在了。回复后的系统将是高效的、简洁的、干净的。好像微软的官方网站上也曾有过利用这种方法修复崩溃系统的思路，但现实中，却偶有修复后，无法进入系统的情况发生。原因肯定是安全相关的，但具体机理却一直不清楚，微软在安全方面的确开放的资料很少很少。

因为这种风险的存在，使得这种修复被灌以了“终极”这个名字，终极的意思就是最终的手段，不到最后关头切勿使用。而有的朋友看漫画看多了，认为终极就是最利害的意思，为此强烈批评狙剑的终极修复其实原理很简单，不配终极这个词，这实在是让人无言以对。

至此，自启动项的内容我们就聊完了，下面我们准备继续聊的就是文件篇，还喜欢这篇文章的朋友们，还请多支持多转载，毕竟写这个东西是很辛苦的，能让更多的人看到，也算充分发挥了其价值，也算我的熬夜辛苦没有白废，大家的支持就是我继续下去的动力。

最后再习惯性的加上一段：此文作者为MuseHero您可任意在网络上传播与转载，但请保持完整性，并标明作者与出处。

木马查杀深度剖析之文件篇(一)

五、木马的查杀之文件篇

文件无疑是计算机的根本，数据的最终形式，其重要性不言可喻。文件涉及到的东西实在太多了，在这里我们只讲与木马查杀有关的部分，想详细了解的请参阅相关资料。

与以往一样，我们先了解一些文件相关的基础知识，然后再聊与木马查杀相关的知识。

1、文件基本知识

首先，我们要明确两个概念“文件格式”、“文件系统格式”与一个机制“文件读写机制”，在此基础上我们再谈论文件的隐藏与查找、保护与删除。

A、文件格式

文件格式是什么呢？我们都知道一个文件名字的组成一般都是由文件名与扩展名组成的，比如：Explorer.exe这个是我们经常看到的程序。为什么我们说他是程序呢？因为他的扩展名是“.exe”，这个扩展名表明了他的文件格式，是exe格式的程序文件，而“exe”其实就是“Execute”这个英文单词的缩写，意思是“执行”，即可以执行的文件。

再比如：“A.bmp”文件，对计算机比较了解的朋友，一眼就可以看出，这是一个图片文件，为什么会是图片呢？因为“.bmp”代表的是一个图片格式，这也是一个英文的缩写“Bitmap”（位图）。

所以，不同的扩展名代表了不同的格式，什么是格式呢？格式就是什么东西放在什么地方，什么地方又放了什么东西的一种规则。比如EXE文件，他的格式与BMP文件就完全不同，EXE文件开始部分是DOS头，前两个字节是“MZ”；而BMP文件的开始部分是位图文件头，前两个字节是“BM”，你用记事本打开一个EXE文件与BMP文件比较一下，看看最前面的两个字母是不是一个是MZ一个是BM呢？当然了，不同的并不仅仅是开头，全部都是不一样的。为什么我们要讲这些呢？这是为了告诉你一个事实，那就是“扩展名代表了一个文件的格式，但更改扩展名并不会使文件格式发生改变。张三就是张三，你硬让他改名为李四，也不会改变他是张三的事实”。

你如果把一个程序的扩展名改成了“.bmp”那么这个程序看起来就和图片一样了，图标也会变成图片一样的图标。但他实质上仍然是一个可以执行的程序。当然了，改了扩展名后，你双击它，就无法执行了，为什么呢？因为Windows会认为这是一个图片，嘿，系统是只认名字不认人的。所以，系统会调用图片浏览程序来打开这个伪装的程序，结果显然是失败的，因为这并不是一个图片。但是，如果我们把BMP的文

件关联改成 Explorer.exe 呢？那系统就会调用 Explorer.exe 来打开这个文件，结果呢？就执行了。关于文件关联的情况，请参阅自启动项篇的相关章节。

有朋友可能遇到过，有的木马扩展名并不是 exe 而是 pif、scr 等等，但同样可以用鼠标双击来执行。现在是不是已经明白他们为什么能够执行了呢？呵，不明白我就再总结一下：

一个文件是否可以通过鼠标双击来执行，与其扩展名并没有必然联系。有联系的是该文件的文件格式与在系统中注册的打开此格式文件所使用的程序，缺一不可。

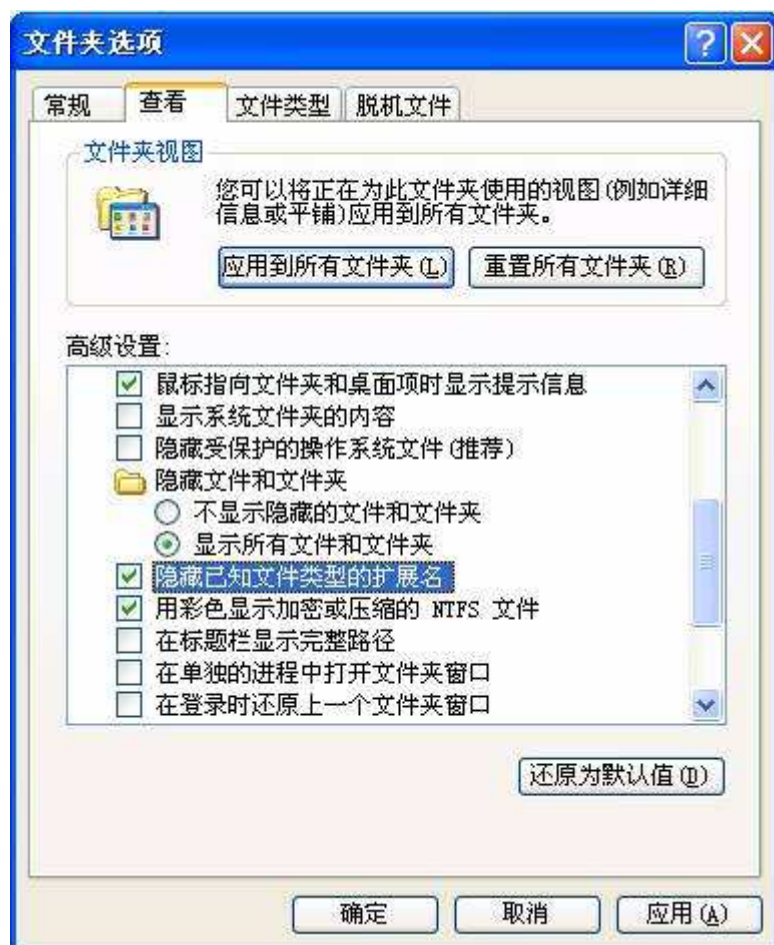
如果将 exe 文件的文件关联改成图片浏览器，那么它也就无法双击执行了。而木马习惯于将自己改成 pif 或 scr 是因为系统中默认的打开这两种格式文件的程序就是 Explorer.exe，与 exe 文件是一样的。

如果指定了用某一程序来打开或加载某一文件，只要文件格式正确就可以正确打开或加载，与扩展名无关。这就是为什么有些进程中的模块看起来并不是 DLL 文件，而有些进程看起来并不是 EXE 文件的原因。扩展名只是告诉系统，这一类的文件需要用某一个程序来打开，如果打开时已经指定了程序，那扩展名就无用了，有用的只是文件格式。

在 Windows 系统中，可以执行的文件格式叫做 PE 格式 (Portable Executable File Format/可移植的执行体文件格式)，包括了 EXE、DLL 等。而可执行文件意指文件中存的是指令序列或与其有关的东西，是可以放到内存中一条条执行的。与其对应的是数据文件，其中存的并不是可执行的指令，而供查看的数据，包括图片、音乐、文档等。

某类型的文件格式会有相应的程序去读取它们，而我们自己也可以定义自己的文件格式，比如：我们可以定义 .SSS 为扩展名的文件格式，并写个程序来读取，这是完全可行的。

这里我们再顺便说一说扩展名的更改，手动更改一是在 DOS 下或控制台下更改，另外就是在文件夹的菜单中选择“工具”→“文件夹选择”，在弹出的窗口中，将“隐藏已知文件类型的扩展名”前面的勾去掉，如下图 03-37：



这样，文件夹中的文件就会显示全部的文件名与扩展名，就可以随便的更改了。

B、文件系统格式

文件格式是指文件中的数据存放的规则，什么地方放了什么东西，而什么东西又放在什么地方。而文件系统格式，则是指文件在磁盘中的存放规则，此规则说明的是文件是如何放到磁盘中的，又如何能找到它们。

我们接触最多的是 FAT 格式与 NTFS 格式，最早的系统比如 DOS、Win3.X 用的是 FAT16，后来的 Win95\Win98\WinMe 开始使用 FAT32 系统，而 FAT32 只是 FAT16 的一个增强扩充版本，并无实质上的格式改变。

NTFS 最早用于 WinNT 系列的操作系统，WinNT4 以后的操作系统都支持 NTFS 文件系统，虽然仍然提供了对 FAT32 文件系统的支持，但 NTFS 取代 FAT32 已经是定局了。而当前仍然有不少朋友在用 FAT32 文件系统，尤其是系统盘使用 FAT32 文件系统的更加普遍。是因为 FAT32 较 NTFS 有某些优势么？非也，据我了解，是因为装机人员出于方便 DOS 下的操作而考虑的，因为 DOS 是不支持 NTFS 文件系统的，这无疑是一种很不负责任的做法。

为什么说这是不负责任呢？这就要从 NTFS 文件系统的优势说起了，当你明白 NTFS 文件系统的优势后，就会明白，为了图一时的方便，用户牺牲的又是什么。

NTFS 系统的优势：

1 容错性：NTFS 可以自动修复磁盘错误而不显示出错信息，在 NTFS 分区写入文件时，系统会在内存中保留一份拷贝，写完后将写入的与内存中的进行比对，如果发现不一致，就将该扇区标为坏扇区不再使用，以免用户数据丢失。

1 安全性：可以设定文件、目录的读写权限，禁止未授权的访问。

1 加密：可以对文件进行加密操作。

1 文件压缩：对不常用的文件自动压缩以节省磁盘空间，支持单一文件或目录的压缩。

1 磁盘限额：对各用户分配可用磁盘空间大小。

1 综合索引功能：使文件查找更快速，这个优势用系统自带的搜索文件功能并不明显，可以用猎剑附带的“磁盘文件搜索”功能，NTFS 文件系统的 30G 的分区十几万文件，搜索完成只需几秒钟；而同样大小的分区与同样数目的文件，如果是 FAT 文件系统，需要的时间则可能是要以几十倍上百倍的计算。

1 可恢复性：系统发生意外出现数据错误时，可以自动进行恢复。

1 其它等等优势……总之 NTFS 比 FAT 有着更高的效率更高的安全性更高的……

除了 FAT 与 NTFS 外，其实我们还经常接触另外的两种文件系统格式，但却不像 FAT 与 NTFS 那么去注意它们罢了，它们就是 CDFS (CD-ROM) \UDF (DVD-ROM)，呵，是不是呢？光盘都接触过吧？不过，我们这下面将只讲 FAT 与 NTFS，因为他们与我们的主题木马的查杀是最相关的。

文件格式有特定的程序来识别并读取使用，而文件系统格式则由特定的驱动来识别并读取使用，这就是我们经常听到的 FSD（文件系统驱动），而 FSD-HOOK 就是 HOOK 这几个驱动了：

CDFS 格式：Cdfs.sys

UDFS 格式：Udfs.sys

FAT 格式：Fastfat.sys

NTFS 格式：Ntfs.sys

我们不用关心，事实上的数据在磁盘中是如何存放的，每种格式的存放方式都不相同，想了解这个的去查阅相关资料，我们现在只需要明白，文件系统格式与文件系统驱动这个概念就可以了。下面呢，我们就看一看，一个文件是如何读取的，了解了原理上的东西，再看问题就会清晰很多。

C、文件读写机制

在了解这个之前呢，我们仍然需要先聊一些必要的基础知识，（咳，没办法，任何东西都不是孤立存在的，都需要一些知识为根基），文件读写离不开缓存管理，我们先看看缓存管理方面的。

从宏观上来说，计算机的数据流是由磁盘流向内存再流向 CPU 的各寄存器，而 CPU 是执行速度最快的，汇编语言之所以被认为是效率最高的语言，很大原因在于使用汇编语言可以直接操作 CPU 的寄存器，可以最大限度的抛开内存与磁盘的读取去利用寄存器。速度其次的，就是内存了，尽力的自内存中读写数据，

而抛开磁盘操作，是无法直接使用寄存器时要努力做到的。而速度最慢的，就是磁盘了，直接从磁盘中读数据，是速度最慢的。

不知大家明白上面说的不？总之，系统在努力的保证更多的操作都是直接操作的内存而非磁盘，以此来提高系统的整体效率。

但是文件是保存在磁盘上面的，避开磁盘读写显然是不可能，只能是尽力的减少读写磁盘的次数。那么，系统又是如何来尽量多的读写内存而不是磁盘的呢？这就是我们接下来要讲的三个概念“缓存”、“智能预读”与“延迟写”。

缓存是什么呢？缓存的用途是将频繁被操作的数据的一个子集保存在物理内存中，这样我们的频繁操作就可以直接操作内存，而不是频繁的读取磁盘了。但系统又如何知道我们要操作哪些数据呢？这就涉及到智能预读的概念了，“智能”？明白了么？是通过一种科学的计算，来决定哪一部分应该被事先读取，并保存在内存中。当我们打开一个很大的文件或频繁的对某一文件进行操作时，系统会预选读取一部分放到内存中，预先读取哪些是通过计算得来的（这种计算的命中率显然决定了性能）。以后我们的读写操作就都是在读取内存，而非速度低下的磁盘。

当打开某一大的文件时，第一次会很慢，但关闭后再打开时会快很多，有这体会么？

如果有 100 本很长的小说，其中 10 本是我们经常阅读的，那么在对这 10 本进行操作时，其速度要比操作另外 90 本明显快上很多，这都是预读捣的鬼。

延迟写又是什么呢？比如我们修改一个文件，看着其中的一个字母“A”非常的不爽，就将它改成了“B”，改完后感觉仍然不够好，就再改成“C”……这种修改如果系统每次都修改的结果存到磁盘上，那么我们的磁盘岂不是时时刻刻都在狂转？所以，系统采用了一种延迟写的方法，我们所有的修改都存在于缓存中，当完成一定的积累后，系统再一次性的将所有修改写入磁盘，而这种写是在后台进行的，我们并没有更多的感觉。

这里需要注意的地方是，这种延迟写与前面自启动项篇中注册表 HIVE 文件的延迟写并不是完全相同的。当然了，原理与机制是相同的，都是为了减少读写磁盘的次数也都是利用了缓存机制。但注册表 HIVE 文件的延迟写是由一个专门的注册表文件操作线程负责的，每 5 秒触发并写入一次。而文件的延迟写是由缓存管理器的延迟写出器在一个系统辅线程上每秒钟一次的来执行的，而且，执行并不等于写入，执行一次意指计算一次，计算什么呢？计算是否需要写入磁盘写入多少又写入哪些。

我们抛开缓存管理只谈文件读写，那么其机制就是这样的，系统得到文件读写请求后将请求发送给相关的服务处理程序，服务处理程序则交给文件系统驱动，文件系统驱动进行安全检查，看我们是否有权力进行所要求的操作，如果通过，则首选自缓存中取数据给我们，只有当缓存中没有时，才会自磁盘中定位到数据在的位置去取数据。

汗，唠叨了半天只是让大家明白了有这么一个缓存的存在。这一段实在是很难写，因为文件系统是系统中很复杂的一块，涉及到的关联东西很多，写多了篇幅过长且会很杂乱，写少了又很不清楚，也不知道大家能否看明白，实在不明白就翻翻相关资料吧。

2、文件的隐藏、查找、保护与删除

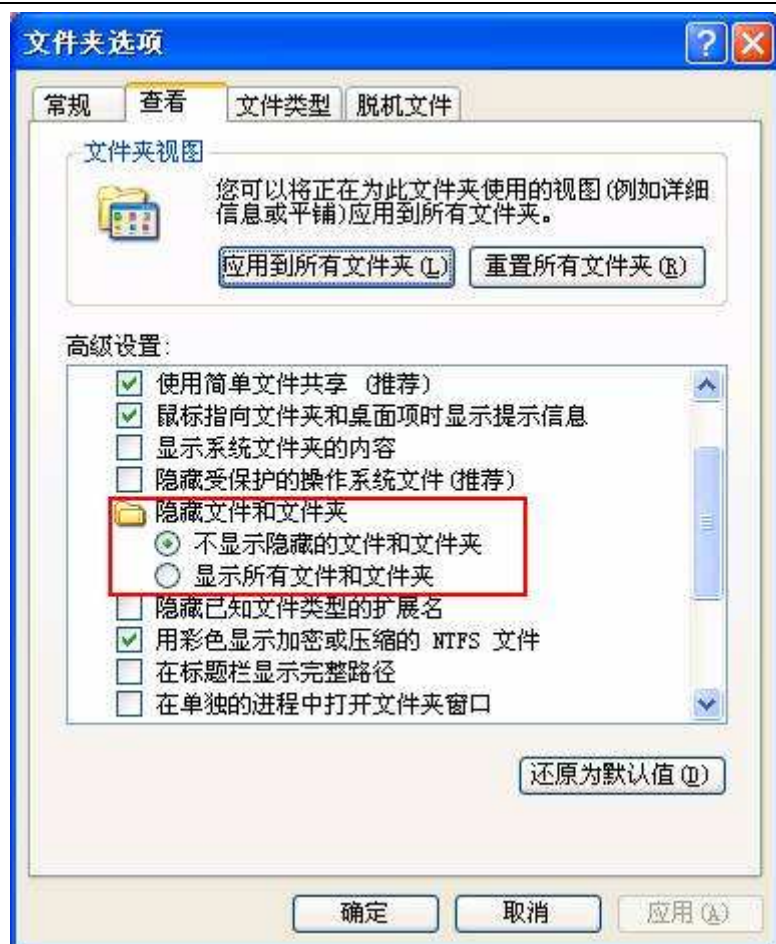
文件的隐藏与保护的方式很多，我们下面对当前主流的一些隐藏及保护的方法进行一下讲解，在说之前，先要明白文件夹是什么，对系统来说文件夹其实只是一个具有特殊属性的文件而已。所以，文件的保护与隐藏同样适用于文件夹，当然了，操作起来由于文件夹的特殊性，会有所不同。我们讲解的顺序是由简到难的，首先讲的就是利用系统本身的功能来隐藏文件或文件夹（为了讲解方便，后面所指的文件也包括文件夹，隐藏也包括保护）。

A、利用系统本身的功能隐藏文件

文件本身的属性中，就有“隐藏属性”，当然勾选了文件的隐藏属性，并设置系统选项为“不显示隐藏文件”时，具有隐藏属性的文件，将不会再显示在文件夹中。

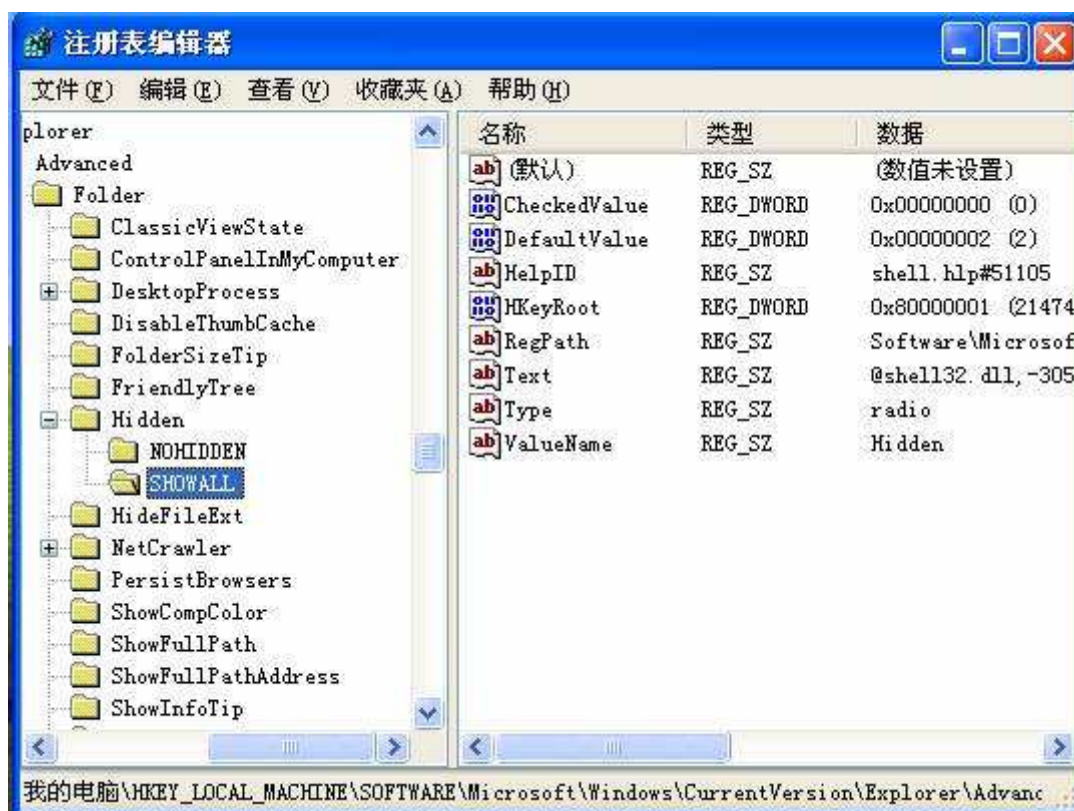
在文件上面按鼠标右键，选择“属性”，在属性设置的最下面勾选“隐藏”，就可以设置此文件为隐藏文件。

然后在文件夹上方的菜单里，依次选择“工具”→“文件夹选项”就可以出现下图 03-38：



将红框中的选项改为“不显示隐藏的文件和文件夹”，点击“应用”或“确定”，就可以完成设置，然后你就会发现，你刚才设置为隐藏属性的文件，已经不见了。

这个设置起来很简单，而简单也就意味着，这种隐藏几乎是没有什么用的，因为其它人也可以通过这种方法来选择“显示所有文件和文件夹”将你隐藏的文件重新显示出来。有没有办法加强这种方法的效果呢？答案是肯定的，有木马就用的这种方法的增强版，而效果看起来还不错，至少有很多人就拿它无可奈何。如何增强呢？我们看下图 03-39：



如果你是把这一系列教程从头看到了尾，那么很容易就知道，上面是注册表编辑器。是的，我们需要在注册表中进行设置，设置的键为：

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL

选中此键后，再在右侧的设置窗口中，将“CheckedValue”这一项的值改为“0”，就完成了增强版的文件隐藏设置。会有什么效果呢？

我们再打开 03-38 图中的文件夹选项设置窗口，然后选中“显示所有文件与文件夹”，选“确定”，去文件夹中看看，刚才我们设置为隐藏的文件能看到不？是不是仍然看不到？这是咋回事？再次打开文件夹选项设置窗口，赫然发现，我们刚刚设置成的“显示所有文件与文件夹”又被自动的设置成了“不显示隐藏的文件和文件夹”。

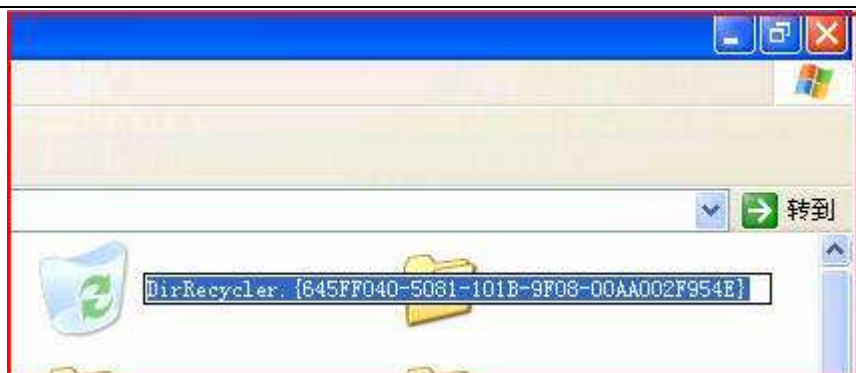
呵，记得当年好像是“橙色八月”这个木马最先用的这种方式来隐藏文件，很有创意是不？

木马查杀深度剖析之文件篇(二)

A、利用系统本身的规则隐藏文件

接下来我们要讲的文件隐藏方法跟上一种类似，同样为无技术含量的障眼法，但这种方法当前却被大量的文件夹加密、隐藏类的软件利用，更有很多商业软件都在用这种方法，而且据说卖的还不错。唉，看来技术跟市场的确是不能划等号的。

我们先看下图，下图是某一文件夹隐藏软件隐藏一个文件夹后的结果 03-40：



文件夹是被隐藏了，可根目录下多出了一个类似回收站的东西（有的干脆就用了系统的回收站，这个是自己新建了一个回收站）。

回收站可以创建吗？呵，其实回收站就是一个特殊的文件夹而已，关键是后面的扩展类名。看上图中文件名字后面的那个“{645FF040-5081-101B-9F08-00AA002F954E}”。我们自己随便新建一个文件夹，然后重命名，在原名字后面加上{645FF040-5081-101B-9F08-00AA002F954E}，我们建的这个文件夹也就成回收站了。改回去也很简单，将后面那一堆去了，就 OK 了。

这里附上一堆这个东西，大家闲着无聊的可以改着玩一玩：

- . {21EC2020-3AEA-1069-A2DD-08002B30309D} 控制面板
- . {2227A280-3AEA-1069-A2DE-08002B30309D} 打印机
- . {D6277990-4C6A-11CF-8D87-00AA0060F5BF} 任务计划
- . {645FF040-5081-101B-9F08-00AA002F954E} 回收站
- . {7BD29E00-76C1-11CF-9DD0-00A0C9034933} 历史文件夹
- . {871C5380-42A0-1069-A2EA-08002B30309D} IE
- . {208D2C60-3AEA-1069-A2D7-08002B30309D} 网上邻居
- . {992CFFA0-F557-101A-88EC-00DD010CCC48} 拨号网络

当然了，仅仅改成个回收站显然是不够的，我们看看它在里面还玩儿了什么花样，将回收站后面的扩展名去掉，然后打开文件夹，就会看到里面还有两层文件夹，如下图 03-41 所示：



首先，里面是一个名字为“nul”的文件夹（这个名字也是有学问的，现在我们先不管它，后面会提到），再打开 nul 文件夹，里面是一个名字为“userfiles.”的文件夹，再打这个文件夹时，完了，打不开了，会弹出一个如上图所示的对话框。尝试删除之，也是失败！

嘿，我们去控制台看看，是否可以进去逛逛呢？

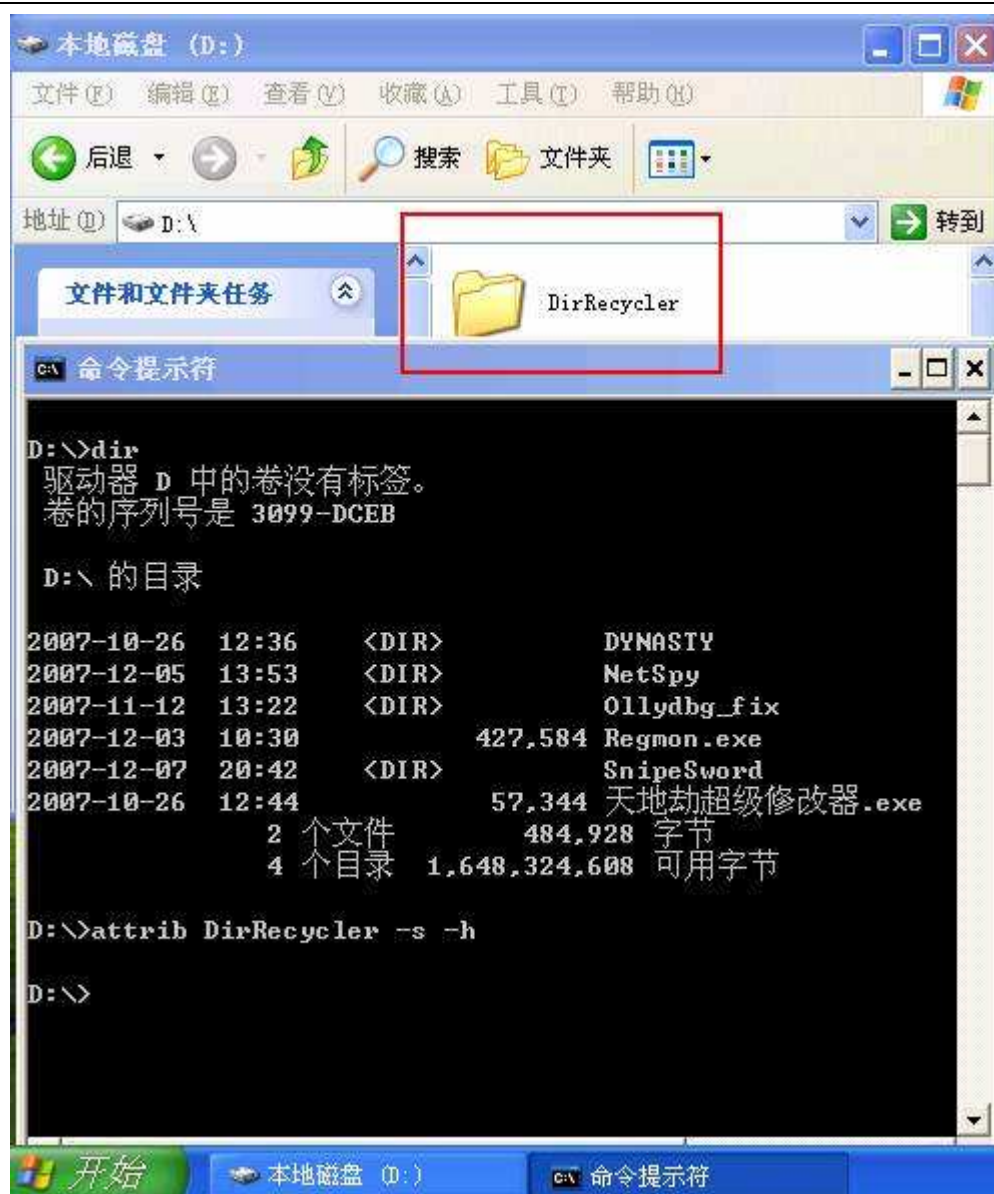
进入控制台，输入“dir”列出 D: 下的文件与文件夹，嗯？居然没有那个文件夹？看下图 03-42：



隐藏了么？按我们上面说过的方法，看一看，在文件夹上按右键，选属性，果然是隐藏的，不过……是灰色的，不能改回去，如下图 03-43：



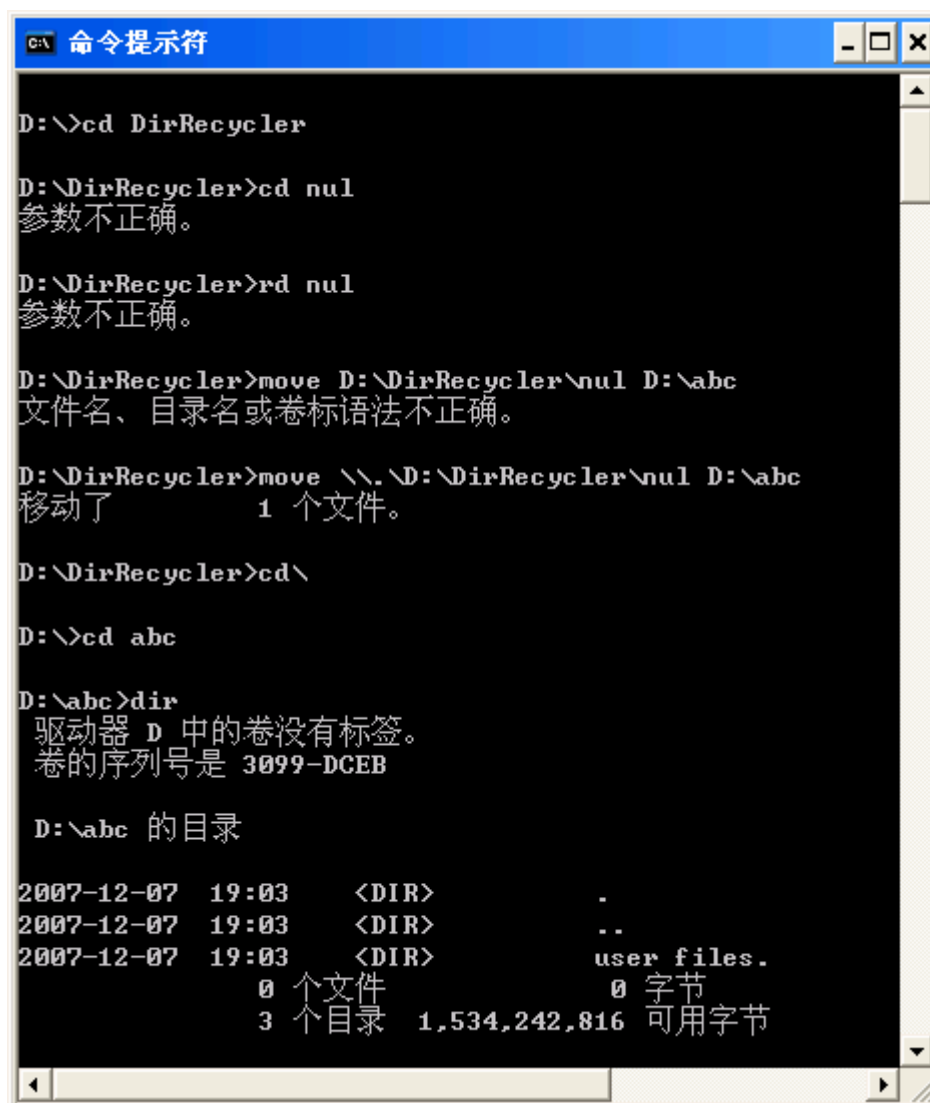
当然，虽然我的水平不高，这一点还是难不倒我的，不是已经进入控制台了么？改个属性很容易吧？我来改改看，看下图 03-44：



输入“attribDirRecycler -s -h”，回车，再看文件夹中的文件，上图红框圈起来的地方，文件夹是不是已经鲜亮起来？不再是隐藏后的灰色？呵，再确认一下后，确定，隐藏属性已经被去掉了，再输入“dir”，OK，它出现了。那我们输入的那一行是做什么用的呢？“attrib”是一个DOS命令，是更改文件及文件夹属性的，紧跟在后面的“DirRecycler”就是我们要更改属性的文件或文件夹，再后面的“-s”是告诉系统，我们要去掉DirRecycler的系统属性，为什么要去掉“系统属性”呢？因为系统属性才使得我们无法在Windows的文件属性里更改其属性（使更改隐藏属性的选项变成了灰的）；再后面的“-h”是告诉系统，我们同时还要去掉它的隐藏属性。如果把

-s-h换成+s+h就是添加系统与隐藏属性了。

好了，既然已经看到了，我们就进去看看吧。CDDirRecycler是进入DirRecycler文件夹，同样的CDnul就是进入nul文件夹了，看下图03-45：



```
C:\>命令提示符

D:\>cd DirRecycler

D:\DirRecycler>cd nul
参数不正确。

D:\DirRecycler>rd nul
参数不正确。

D:\DirRecycler>move D:\DirRecycler\nul D:\abc
文件名、目录名或卷标语法不正确。

D:\DirRecycler>move \\.\D:\DirRecycler\nul D:\abc
移动了 1 个文件。

D:\DirRecycler>cd\

D:\>cd abc

D:\abc>dir
驱动器 D 中的卷没有标签。
卷的序列号是 3099-DCEB

D:\abc 的目录

2007-12-07 19:03 <DIR> .
2007-12-07 19:03 <DIR> ..
2007-12-07 19:03 <DIR> user files.
0 个文件 0 字节
3 个目录 1,534,242,816 可用字节
```

进入 DirRecycler，没问题，进去了。再进入 nul，不让进了，参数不正确，为什么不正确呢？嘿，因为 nul 是系统的设备名字，进入设备，当然会被拒绝了，这回知道他为什么要将文件夹取名为“nul”了吧？人家就防着我们在控制台进入呢。

嘿，恼了，删除它。Rd 是删除文件夹的命令，失败，呵，删除设备？开玩笑呢吧？move 是移动文件夹（间接有重命名的意思），我们给它改个名字成不成呢？仍然失败，给设备改名字，有没有搞错？汗，看来系统还挺死心眼的。

换个方法，看上图：在路径前加上了个“\\.\”，哈，成功了！文件夹 nul 成功的被移出了 DirRecycler，并改名为“abc”，进入 abc 看一看，果然没问题，里面就是那个 userfile. 文件夹。

“\\.\” 是什么东西呢？这是 UNC（通用命名约定）路径的标志，是一种网络路径，我们在这里用这种路径其实就是告诉系统这个死心眼的家伙，这不是什么设备只是个文件夹而已。

OK 了，终于将 nul 给搞定了，其实系统中并不仅仅 nul 是设备名，AUX、COM1、CON、PRN……等等还有很多，当发现类似情况时都可以用这种方法处理。

我们接着进入“userfile.”，晕，还是进不去。恼了，这次是真的恼了。（呵，其实我在没移动 nul 文件夹前就已经恼的不行了，动手将整个文件夹给删了。）看下图 03-46：

```

C:\>命令提示符
D:\>attrib DirRecycler -s -h

D:\>dir
驱动器 D 中的卷没有标签。
卷的序列号是 3099-DCEB

D:\>的目录

2007-12-07 21:13 <DIR> DirRecycler
2007-10-26 12:36 <DIR> DYNASTY
2007-12-05 13:53 <DIR> NetSpy
2007-11-12 13:22 <DIR> Ollydbg_fix
2007-12-03 10:30 427,584 Regmon.exe
2007-12-07 20:42 <DIR> SnipeSword
2007-10-26 12:44 57,344 天地劫超级修改器.exe
                2 个文件 484,928 字节
                5 个目录 1,648,324,608 可用字节

D:\>rd /s \\.\D:\DirRecycler
\\.\D:\DirRecycler, 是否确认(Y/N)? y

D:\>rd /?
删除一个目录。

RMDIR [/S] [/Q] [drive:]path
RD [/S] [/Q] [drive:]path

    /S    除目录本身外, 还将删除指定目录下的所有子目录和
           文件。用于删除目录树。

    /Q    安静模式, 带 /S 删除目录树时不要求确认

D:\>
  
```

这是我更改完文件属性后，直接用 rd+UNC 路径的方式，将整个文件夹全删除了，删除时里面的文件夹并没有移出来，/S 的意思上图中显示的已经很清楚了是删除全部子目录，所有的 DOS 命令都可以用：命令+/?的形式显示帮助信息的。

也就是说，我们根本无须关心文件夹里面是什么，就可以直接删除这个文件夹，上面唠叨了一堆，其实只是为了给大家讲一讲怎么对付用设备名字当文件夹名的，并熟悉一下常用的而又比较有用的几个 DOS 命令而已。

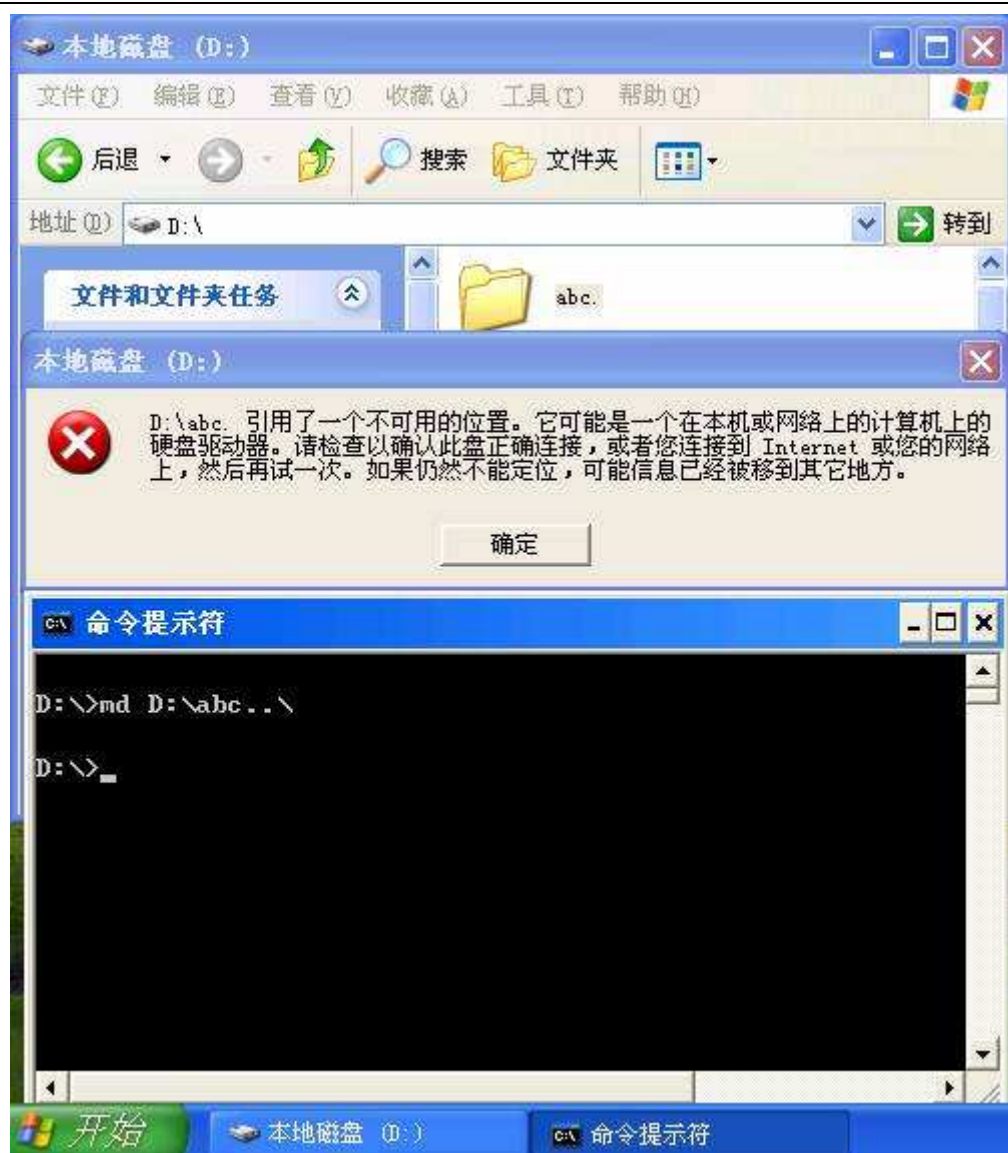
现实中，删除这个是很容易的，但是如果文件夹里面的东西很重要，我们还要取出来怎么办呢？也很简单，看下图 03-47：



在运行里面输入完整的路径，如上图所示，点确定，就可以轻易的打开被重重保护的文件夹，任意操作里面的文件！

是不是简单的可怕？商业软件采用这种方式来隐藏文件夹会不会让你有种发狂的感觉？

Userfile. 这个目录比较特殊在隐藏中也起到了关键的作用，他其实是上面路径中的“userfile..\”，想不想自己也创建一个这样的文件夹呢？看下图 03-48：



在控制台输入“`md D:\abc..\`”，就在 D: 根目录，创建了一个“abc.”的特殊文件夹，试试效果，双击打开那个 abc. 文件夹，系统就傻乎乎的弹出了一个框框。

删除这个文件夹，只需要输入“`rd D:\abc..\`”就 OK 了。

想在里面进行操作，就在运行里输入“`D:\abc..\`”来打开。

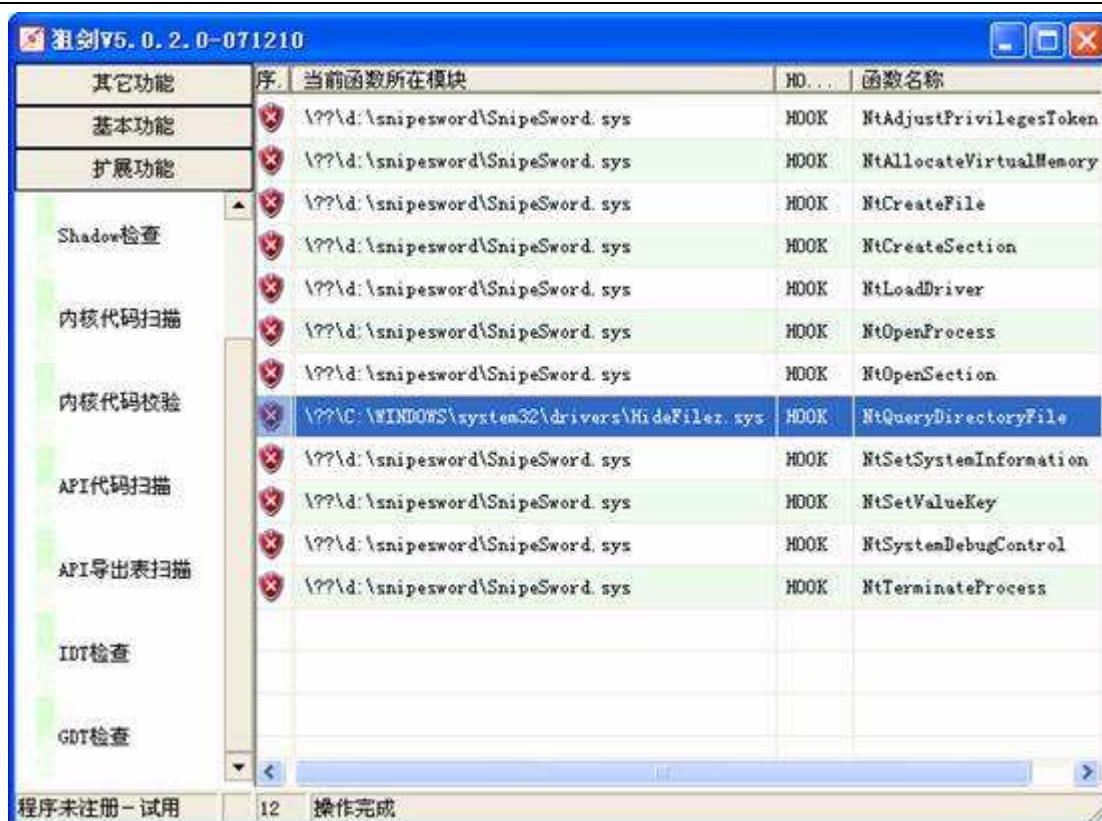
还有一些类似的障眼法，这里就不一一讲解了，下一节里我们要讲一讲采用技术手段进行的文件隐藏。

木马查杀深度剖析之文件篇(三)

A、利用技术手段隐藏文件

讲完了障眼法后，我们讲点实在的。其实，如果您是从第一篇一直看过来的，那么应该可以想到一些手段了，是的，HOOK、Inline-HOOK 几乎可以用于任何地方，不但适用于隐藏进程、隐藏注册表，同样适用于隐藏文件，不同的只是 HOOK 的地方不同而已。

看下图 03-49：



这是某文件夹隐藏软件的 HOOK 情况，当用其隐藏文件或文件夹后，可以在 SSDT 检查中发现这个 HOOK，HOOK 的功能函数是“NtQueryDirectoryFile”，当你恢复了此 HOOK 后，被隐藏的东西自然也就显现了。同理，Inline-Hook 此函数，效果也是一样的。

对于 HOOK，每篇都在讲，这里就不过多讲述了。测试程序也不提供了，毕竟人家是商业软件，是*这个吃饭的。说明一下儿，内核代码扫描是扫描并恢复 Inline-HOOK 的地方。

下面我们讲一讲借助文件系统隐藏文件的情况，上面的基础知识中我们已经讲过了，回顾一下儿“文件格式有特定的程序来识别并读取使用，而文件系统格式则由特定的驱动来识别并读取使用，文件系统驱动根据文件系统的不同而不同，FAT 文件系统、NTFS 文件系统分别由 FastFat.sys 与 NTFS.sys 来管理。”

这里我们再讲一个概念“文件过滤系统驱动”，听名字应该也能想出大概了吧？是的，这是一个文件过滤系统，付在文件系统上，像个筛子一样对传入文件系统驱动的信息进行过滤。文件过滤系统在现实中应用非常广泛，几乎所有的杀毒软件都会用到，而系统本身的系统还原也是应用的文件过滤系统，还有些硬盘还原之类的程序也在使用文件过滤驱动。

在网上找到了个借助文件过滤系统驱动来隐藏文件的例程，这里我们就借由此程序来看看文件过滤系统是如何隐藏文件的。（在此说明一下，说来也惭愧，测试程序都是在网上找的，其实应该是自己来写的，只是我的时间实在是太不够用了，只好偷懒了，在此也谢谢这些程序的作者们。）

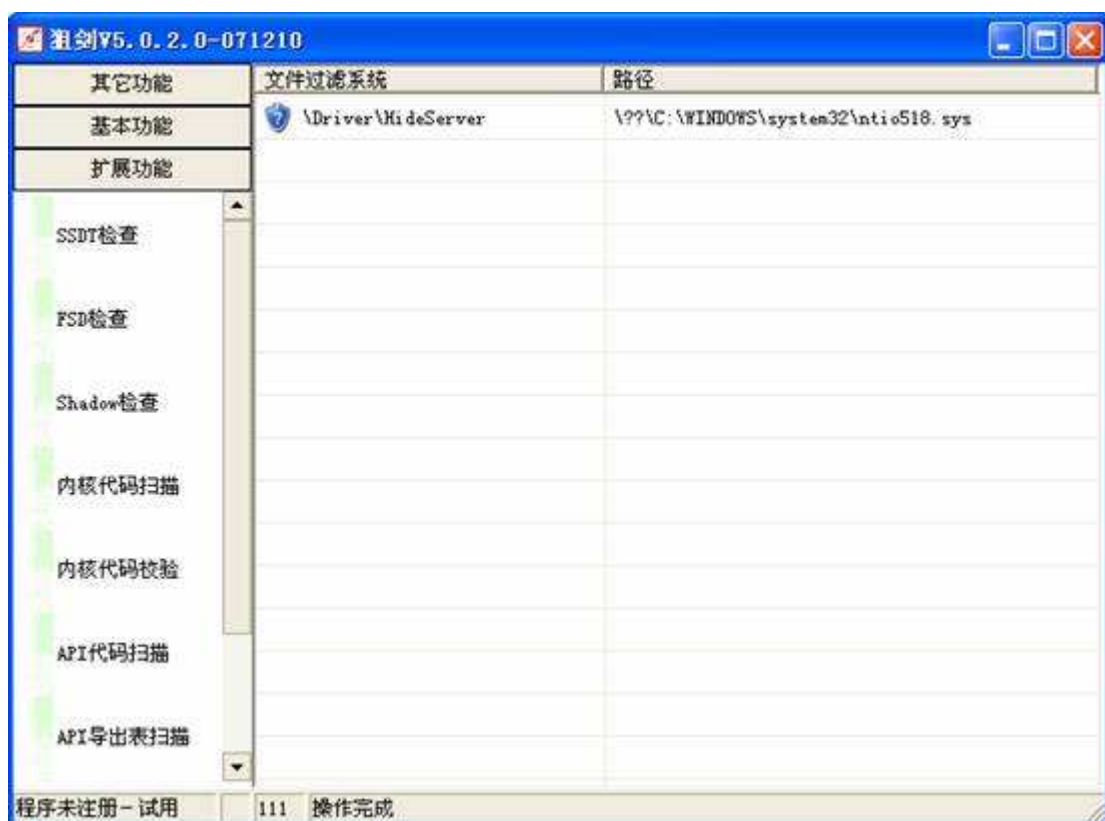
测试程序下载地址：<http://www.zhulinfeng.com/Download/HideFile1.rar>

里面是有一个主程序，两个批处理文件。里面有使用说明，这里就不细说了。先看下图 03—50：



运行 hideA.bat，同目录的 HIDE.exe 会被隐藏，这时就可以用各种工具来测试了，看是否能看到这个文件。运行狙剑的磁盘文件管理，在弹出的选择窗中选择 C: 盘，找到 Test 文件夹，如上图所示，会发现，里面有一个名字是 HIDE.exe 的被隐藏的文件。

从哪里可以看到它是怎么隐藏的呢？文件系统过滤驱动又如何查看呢？看下图：



在狙剑的扩展功能里，有 FSD 检查，FSD 就是文件系统驱动，而在 FSD 检查列表中按右键，里面有一项是“文件过滤系统检查”就不可以查看本机安装的文件过滤系统了。（注意，请使用最新版的狙剑，旧版的虽然可以看到文件过滤系统，但显示文件路径有问题）

这里没提供御载的功能，但是这类驱动都可以通过在“自启动项管理”中找到并清掉其启动项后，重启计算机来清除。

文件过滤系统驱动之下，就是文件系统驱动了，HOOK、Inline-HOOK 文件系统驱动都可以隐藏文件或保护文件不被删除。对文件系统驱动的 HOOK 比文件过滤系统更深，由于在网上没找到相应的程序，这里就不过多讲解了。在“图文详解高级木马的自我保护与查杀之策”中，有对这一块进行过详细的说明，不了解的可以去看看。

其实，原理都是一样的，就像我在前面讲到的一样，当我们向操作系统发出指令后，操作系统有一个由外向里传递指令，再由里向外返回结果的过程，参于这一过程的有很多环节，在任何一个环节被拦截，都将导致我们的请求得不到执行或返回一个错误的结果，而安全软件的目的，就是尽量的减少这个环节，最好是由我们发送指令开始，省略掉所有的中间环节，直接得到结果。

以狙剑的磁盘文件管理来说吧，为什么他可以看到被文件过滤系统隐藏甚至被文件系统驱动 HOOK、Inline-HOOK 隐藏的文件呢？就是因为他最大化的减少了这个指令执行的环节，自己实现了文件系统的解析。

什么又是自己实现文件系统的解析呢？我们已经讲到，文件系统格式是文件在磁盘中存放的规则，文件存在了磁盘的哪里，磁盘的什么地方存放了什么东西只有相应的文件系统才知道，而自己解析就是自己分析磁盘扇区找到文件存放的规律，然后绕过文件系统驱动，自己去磁盘上找文件，并列出来，这样，中间各个环节的过滤与 HOOK 就起不到作用了。

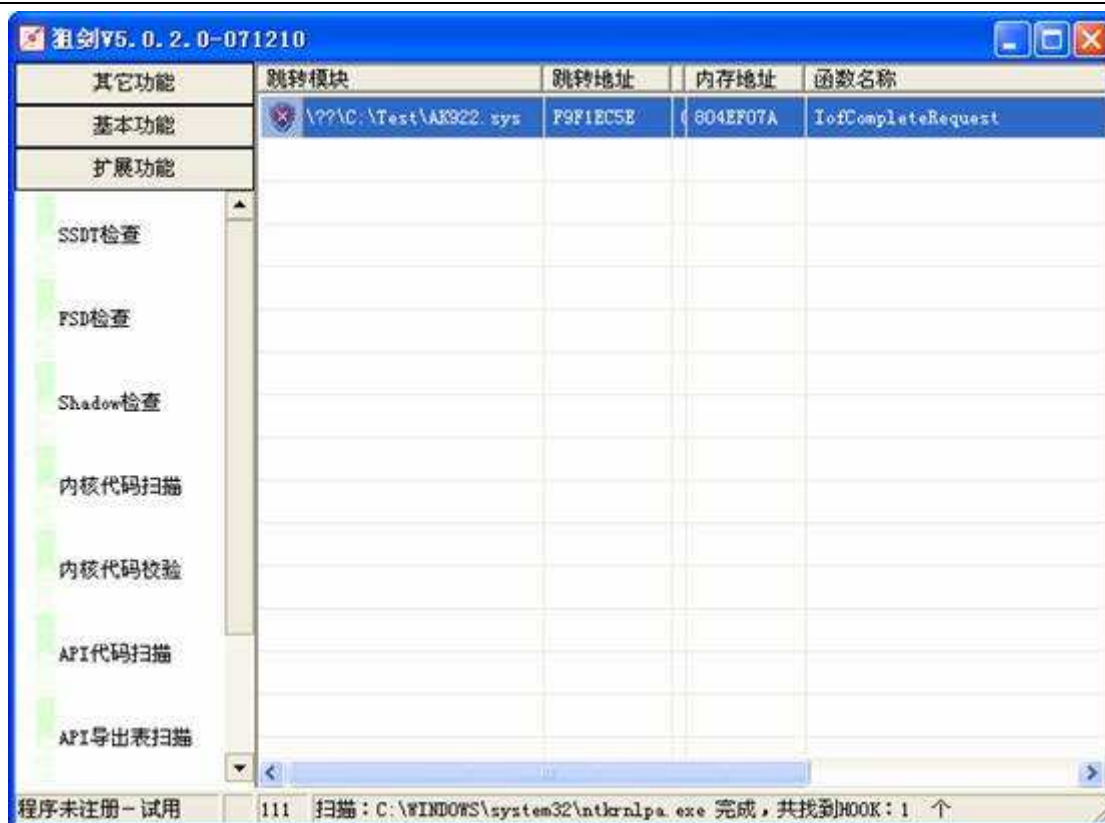
文件的删除也是同样的道理，是否允许删除是由系统在 FSD 环节中检查的，而狙剑的破坏文件是从发出破坏的指令开始，直接跳到了抹杀硬盘上的文件数据，自然中间环节中是否允许的检查也起不到作用了。

那是不是自己实现文件系统解析就可以高枕无忧了呢？也不是，计算机中的一个操作涉及到方方面面，很难完全绕过，绕过了这里又触动了那里，如果有人针对文件系统解析来作手脚，也是完全可以的。再提供一个测试程序：

<http://www.zhulinfeng.com/Download/HideFile2.rar>

这个也是从网上找到的，里面有详细的测试使用说明，这里也不多说了，只强调一点，测试程序没经过严格测试，很不稳定，有可能会蓝屏，我试用了三次其中蓝了一次。

这个程序的作者专门针对自己解析文件系统的安全程序做了有针对性的 HOOK，我们看下图 03—52：



当你装载隐藏文件的 AK922.sys 驱动，并成功隐藏文件之后，可以运行狙剑的“内核代码扫描”检查一下，会发现如上图所示的 HOOK。他就是因为 HOOK 了这个“IoCompleteRequest”才使他成功的隐藏了文件，包括在自己解析文件系统的安全程序中隐藏其文件。（注：呵，你不用再用狙剑的磁盘文件管理来查看这个文件了，你只要一运行狙剑的磁盘文件管理，那么，你在文件夹中直接就可以看到它了。因为狙剑的磁盘文件管理默认会自动的恢复这个 HOOK。）

呵，上面说过了，安全软件检查时很难绕过所有环节，绕过了这里又触动了那里，为木马提供了可乘之机。而木马又何尝不是如此？隐藏了这里又露出了那里，为安全软件的检查提供了线索。所以，检查时多方面就行检查，才是最完善的，不要仅查一点。

其实，这就跟我在进程的隐藏中说到的一样，这是一个不断斗争的过程，没有绝对的胜出者，木马隐藏与保护的新方法层出不穷，而安全软件的检查方法也在不断的更新，即使是昨天的技术可能就已经不足以检查出今天的木马，“不断的更新”是一个安全软件保持其有效性的最佳手段，而经常的翻新也是木马生存的基本条件，就看谁更努力了。

遗憾的是从现在国内的软件行情来看，木马作者的更新动力比安全软件作者的动力要大上很多，巨额利润的引诱不但带给了其不断钻研创新的动力也给了其安心研究的物理环境，而安全软件作者的不断努力更多的却仅来自于责任心跟荣誉感，白天为了生活摆地摊卖白菜，晚上加班跟木马较劲。呵，不知这种生活能坚持多久。更不知道那些大叫着我只用免费软件的人，是如何想的，我想他们所说的免费软件也包括木马吧，毕竟木马应该是永远不会向使用者收费的。

好了，不说废话了，文件篇到此也就结束了，是不是感觉后面的几章不如前几章详细呢？我也有这种感觉，这一段时间天天熬夜实在是太累了，文章质量不但下降了，其它工作也受到了影响，所以，后面的主动防御篇就向后推一段时间再写，但肯定是会写完的。

这一篇没做检查，发现有什么错误的，或有什么问题，请到论坛留言。另外，如果认为有什么应该讲一讲却遗漏没讲的，也可以提出来，最终我会把所有的文章再作最后的修改与补充，然后制成完本的电子文档放上去，供大家下载。

OK 了，就这样吧，我要早点休息了，明天还要打早摆摊卖白菜呢～^-^